Reinforcement Learning Analysis in Text Flappy Bird

Sofia Bouaila[1]

Centrale Supélec
`sofia.bouaila@student-cs.fr`

Abstract. This paper presents a comparative analysis of Monte Carlo and Sarsa(λ) reinforcement learning agents in the Text Flappy Bird environment. Through systematic experimentation, we evaluate convergence properties, sensitivity to hyperparameters, and transfer learning capabilities. We found that the Sarsa(λ) agent is faster then the MC agent. The notebook is accessible here [4].

Keywords: Reinforcement Learning · Monte Carlo Methods · Sarsa(λ) · Game AI

1 Introduction

Reinforcement learning (RL) has demonstrated remarkable success in game environments, particularly in scenarios requiring real-time decision making. We implement two fundamental RL approaches - Monte Carlo (MC) and Sarsa(λ) - in the Text Flappy Bird (TFB) environment, analyzing their performance characteristics through three key dimensions:

– Convergence dynamics under varying hyperparameters
– Transfer learning capabilities between environment versions
– Generalization across different game configurations

Our implementation follows the theoretical framework established in [1], adapted to address practical implementation challenges in the considered environments, and can be found in [**?**].

2 Methodology

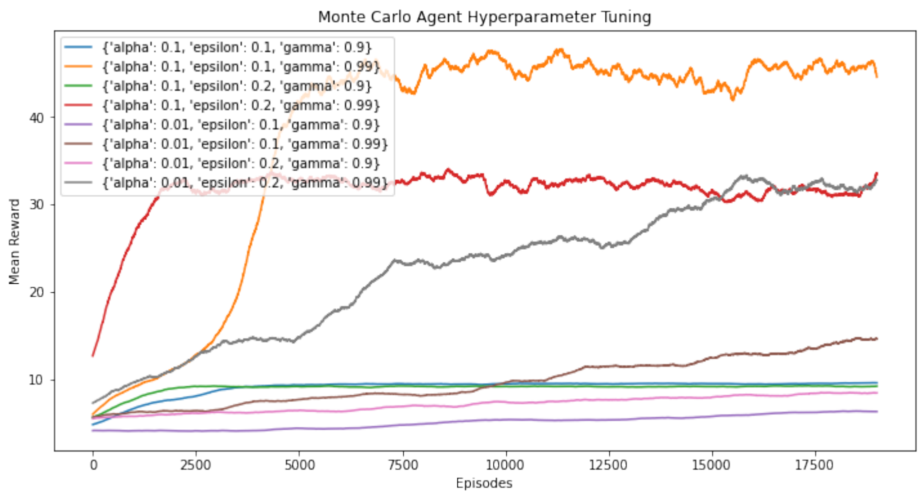
2.1 Environment Specifications

We use the TextFlappyBird-v0 environment with the configuration $height = 15$, $width = 20$ and $pipe\_gap = 4$.

2.2 Hyperarameters tuning

Fine-tuning MC agent

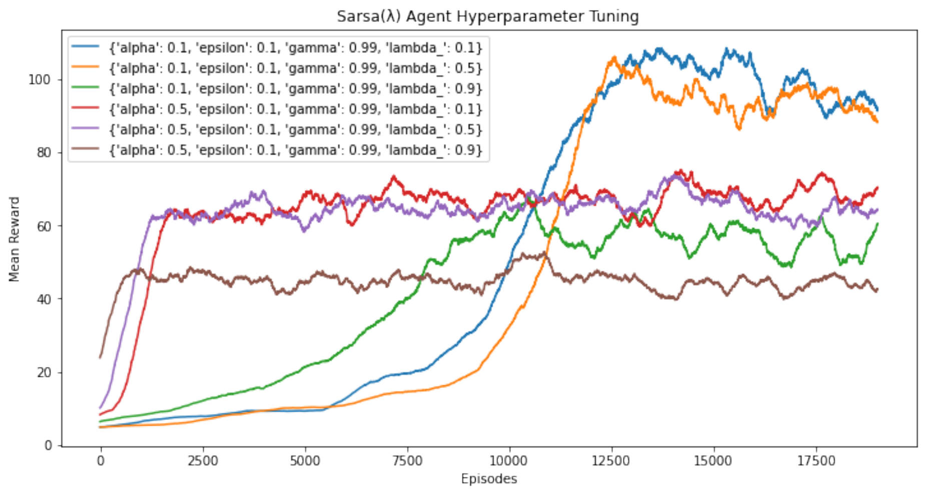
We fin-tune the model using different values for α, γ and ϵ, considering the average reward per 1000 episodes:



This graph shows that the default parameters give the higher reward per episode.

Fine-tunning Sarsa(λ)

We fin-tune the model using different values for α, γ, λ and ϵ, considering the average reward per 1000 episodes:



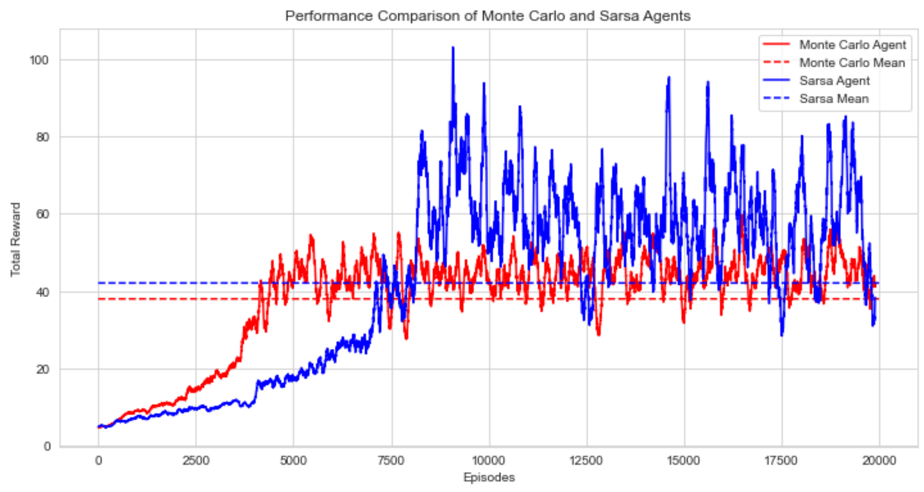
Default Parameters

We use the set of parameters with best results as the default parameters for the agents.

Table 1. Hyperparameter Configurations

Parameter	MC	Sarsa(λ)
Learning Rate (α)	0.1	0.1
Discount (γ)	0.99	0.99
Trace Decay (λ)	N/A	0.1
Epsilon (ϵ)	0.1	0.1

With the default parameters, the MC agent converges towards a stable reward per episode (we do an average of the rewards per 100 episodes), while the Sarsa(λ), even with higher reward average, still needs more episodes to converge.



To compare local behavior of both agent we plot the value function of both agents in the state space, which shows that the reward estimation for almost all the state space is higher with the Sarsa(λ) agent then it is with the MC agent. As well the Sarsa(λ) is most of the time faster then the MC agent.

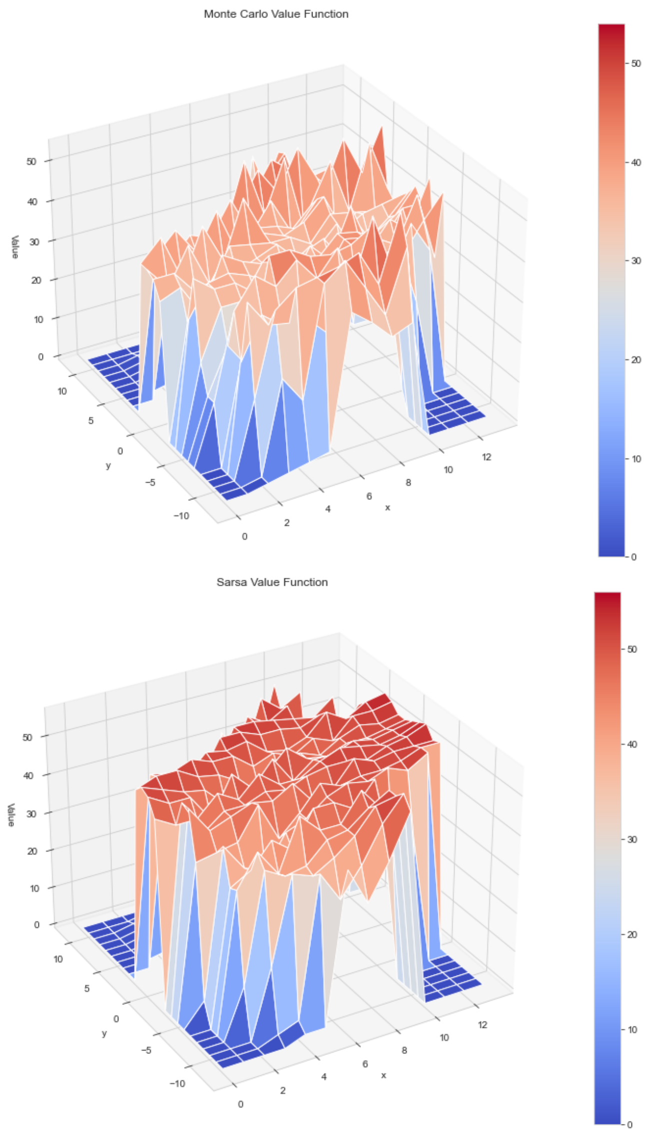


Fig. 1. Value Functions of both agents

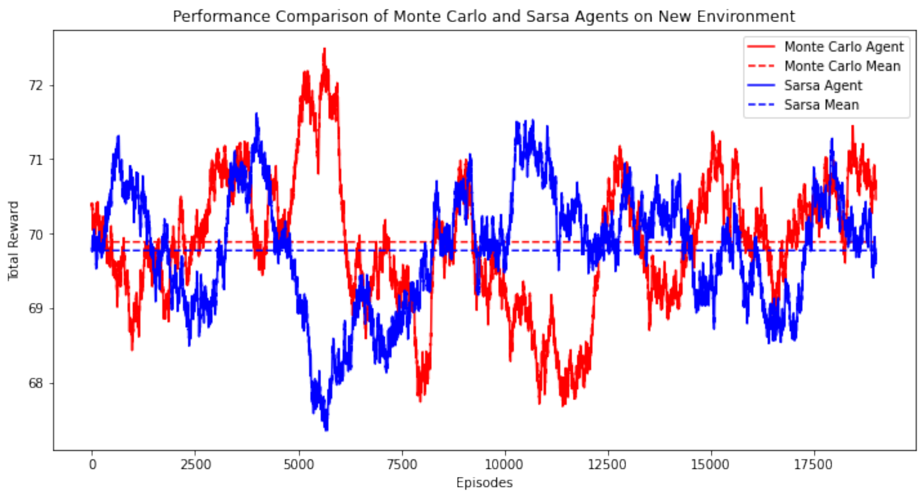
3 Transfer Learning

To evaluate the transferability of our agents, we tested them in the **Flappy Bird Gym** environment [3]. This environment closely resembles the original game, offering continuous state representations and a physics-based engine.

As we implemented the agents to be able to train and take actions in the initial environment and this one, we were able to test the agents on this environment as well.

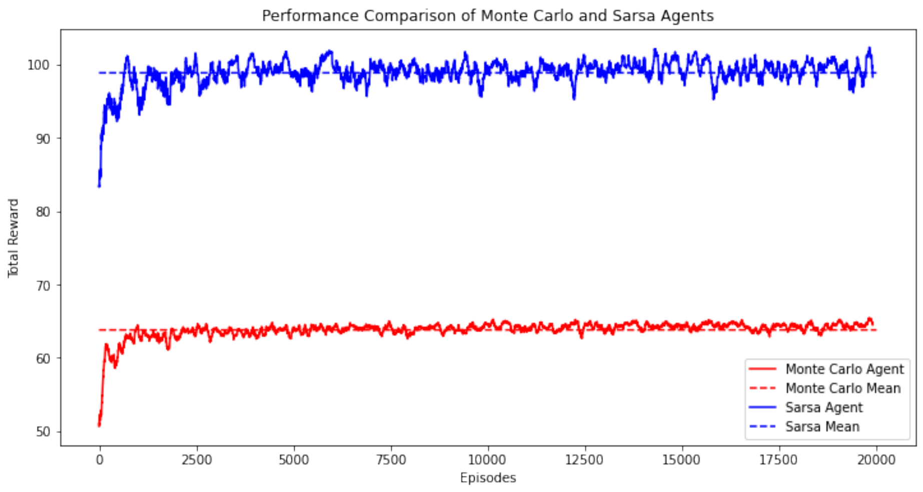
3.1 Using Trained Agents

This graph shows that both agents act in a stochastic way with no clear signs of convergence.



3.2 Training agents in the Flappy Bird Gym Env

After training, we got a clear difference in performance between the 2 agents:



Eventhough the Sarsa(λ) agent is slower then the MC agent but it is more efficient. In fact, the Flappy Bird Gym uses continuous state space which makes it bigger and hard to explore using the Monte Carlo method, instead the Sarsa(λ) agent uses the eligibility criteria which makes it more efficient in larger state spaces.

4 Generalization across different configuration

To evaluate the ability of both agents to generalize across different environment configurations, we use a grid-search across the parameters: *height*, *width* and *pipe_gap*, considering extreme cases.

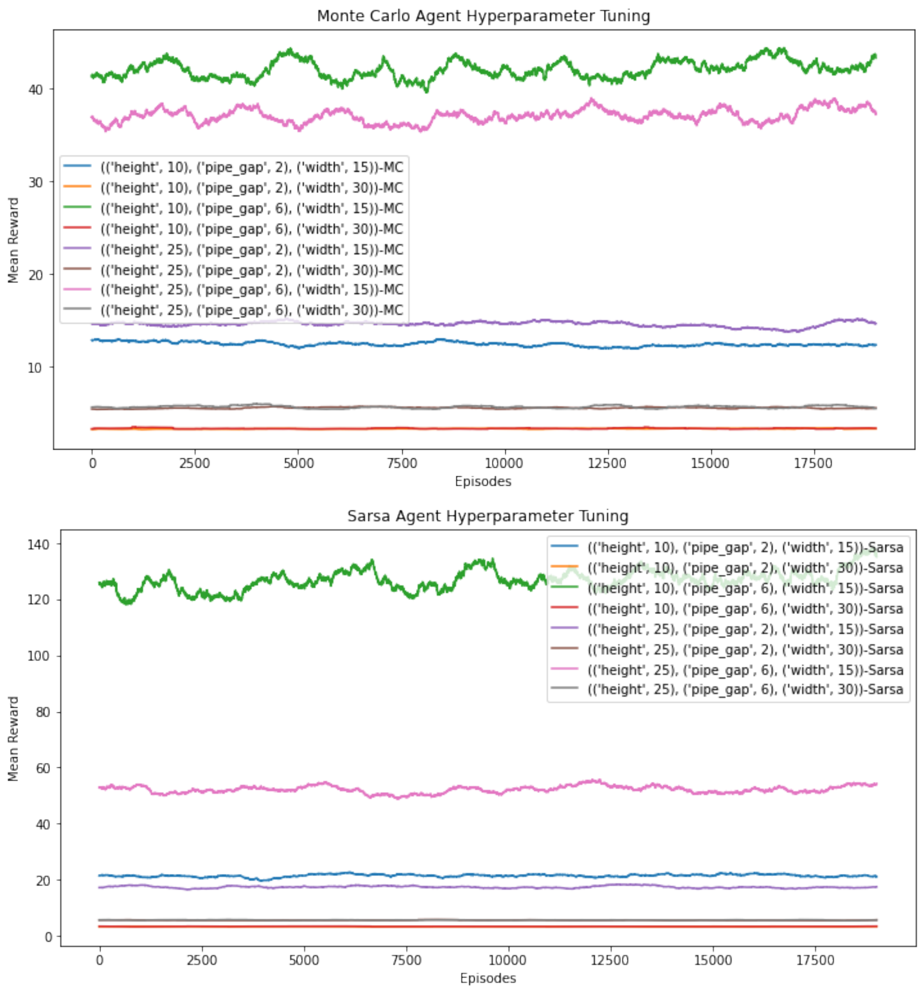


Fig. 2. Agents Performance across different configurations

Observations

- Both models generalize better, Sarsa(λ) the best, when used with midium *width*, large *pipe_gap* and small *height*, which is a simple environment.

- With small *width* we got the best results for both agents.

5 Conclusion

Our analysis demonstrates that Sarsa(λ) is more adapted when using envirnment with continous state space or when generalizing to different configuration, but has higher variance.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. 2nd edn. The MIT Press (2018)
2. Talendar: Flappy Bird Gym Environment. `https://github.com/Talendar/flappy-bird-gym` (Accessed: 2023-09-30)
3. Talendar. *Flappy Bird Gym: OpenAI Gym Environment for Flappy Bird.* `https://github.com/Talendar/flappy-bird-gym`
4. `https://github.com/bosofia/RL-Flappy-Bird/blob/main/assignment.ipynb`