Reinforcement Learning for Text Flappy Bird

Alexandre Boistard

SDI-MMF, Mathematics and Data Science, CentraleSupélec

Abstract. This report investigates the application of reinforcement learning methods to a simplified game environment, Text Flappy Bird (TFB). Two agents-a Monte Carlo-based agent and a Sarsa(λ) agent-are implemented and evaluated on different versions of the TFB environment. The study explores their learning dynamics, sensitivity to parameter configurations, and convergence behaviors. Results indicate that while both methods are viable for solving TFB, each exhibits unique strengths and limitations in terms of convergence speed, stability, and adaptability to environmental changes.

Keywords: Reinforcement Learning, Monte Carlo Methods, Sarsa, Text Flappy Bird.

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for training agents to make sequential decisions in uncertain environments. This report focuses on the application of RL techniques to a simplified version of the popular Flappy Bird game, named Text Flappy Bird (TFB).

The primary objective of this study is to implement and compare two RL approaches-a Monte Carlo-based method and the Sarsa(λ) algorithm-as applied to the TFB environment. The Monte Carlo method provides an intuitive approach to learning by sampling complete episodes and averaging returns, while Sarsa(λ) offers an incremental strategy through eligibility traces, enabling the agent to learn more directly from its experiences.

The report details the experimental setup, including a discussion on parameter sensitivities, convergence behaviors, and performance evaluations. Finally, we examine how a trained agent performs under varied level configurations, shedding light on the transferability of learned policies. Through comprehensive experimental results and analysis, this study aims to contribute insights into the strengths and challenges associated with different RL methodologies in discrete, simulation-based game environments.

2 Selected Models, Experimental Setup

The two versions of the TFB environment present contrasting observation spaces. One version returns a complete screen render of the game, offering a rich, high-dimensional visual input that is closer to real-world scenarios, but at the cost of increased computational complexity and potential difficulties in feature extraction. In contrast, the numerical observation version provides only the (dx, dy) coordinates of the player relative to the nearest pipe gap, simplifying the state representation considerably. While this lower-dimensional input makes learning more tractable and allows for faster experimentation, it may omit useful contextual details present in the full screen render, potentially limiting the agent's ability to generalize to more complex visual environments. **We only focused on the** $(TextFlappyBirdv0)$ **version in this project.**

The training environment was set with a height of 15, a width of 20, and a pipe gap of 4. For the Monte Carlo agent, we evaluated four different learning rates ($\alpha = 0.01$, 0.1, 0.5, 0.9) with a fixed discount factor of $\gamma = 1.0$, and applied an ε-greedy policy with an initial $\varepsilon = 1.0$ that decays multiplicatively by 0.9999 per episode down to a minimum of 0.05. For the SARSA(λ) agent, we investigated multiple (α, λ) combinations, specifically $\lambda \in (0, 0.5, 0.9)$, while keeping γ and the ε settings identical. Each agent was trained over a maximum of 50000 episodes (the convergence was observed around 20000 for both agents) and performance was evaluated by averaging results over multiple runs and by applying bins to ensure robustness and statistical significance.

3 Performance of the agents, parameters influence

The Monte Carlo agent updates its value estimates only after complete episodes, thereby relying on the full return, whereas the SARSA(λ) agent incorporates eligibility traces to update its action-value function incrementally at every step. This key difference leads to distinct learning dynamics: the Monte Carlo approach tends to be more sensitive to the variability in returns and may converge slower, while the SARSA(λ) agent, with its ability to assign credit to multiple past actions, should converge faster and exhibit smoother learning curves. Parameter sensitivity analyses revealed that both agents are very sensitive to the parameter α.

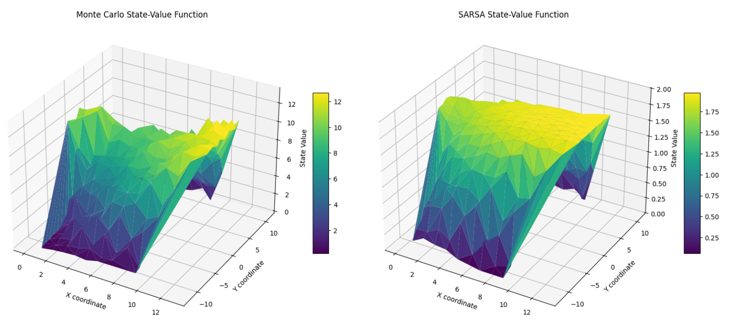

Fig. 1. Monte Carlo and SARSA(λ) state-value functions (5000 episodes)

These 3D plots show that while both agents identify similar high-value regions in the state space, the Monte Carlo surface features sharper peaks, suggesting that its episodic updates can yield more extreme estimates of expected return, whereas the SARSA(λ) surface is smoother, reflecting the incremental updates and eligibility traces that propagate value information continuously. Monte Carlo's higher peaks may indicate some overestimation in states that occasionally produce large returns, whereas SARSA(λ) demonstrates more conservative but stable estimates due to its step-by-step learning process.

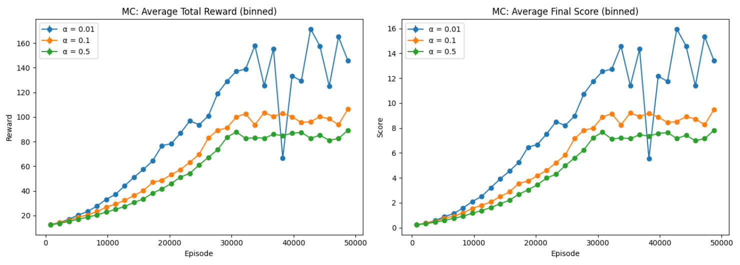

Fig. 2. Monte Carlo parameter sensitivity and convergence time

We now compare with Sarsa:

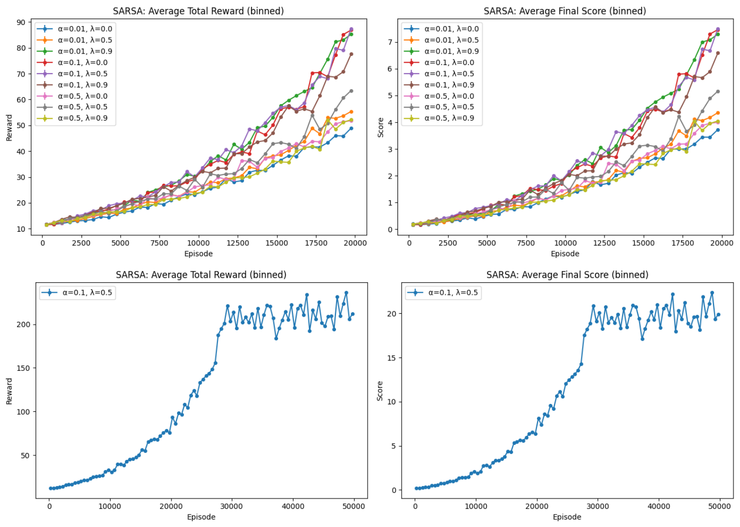

Fig. 3. Sarsa parameter sensitivity and convergence time

These results show that both Monte Carlo and SARSA(λ) exhibit high sensitivity to their configuration. Both seem to converge at around the same rate, but SARSA(λ) converges higher and seems to be more stable (recall that the state value function was smoother, which may explain it, as the MC method find highly rewarded paths, which introduce variance). Overall, Monte Carlo may reach high returns if tuned well but is more sensitive to episodic variance, whereas SARSA(λ) often achieves steadier improvement thanks to its step-by-step updates with eligibility traces.

4 Generalization Abilities

Applicability to the Original Flappy Bird Environment. Although our agents were designed for the simplified TFB environment, the underlying methodologies are general enough to be applied to the original Flappy Bird game. However, transferring the agents would likely require additional preprocessing steps for handling high-dimensional visual inputs, and possibly a re-design of the state representation.

Performance on Different Level Configurations. Evaluations indicate that an agent trained on a specific TFB configuration (e.g., height = 15, width = 20, pipe gap = 4) generally performs best in that environment, achieving higher average total returns and scores. The results indicate that while the agent performs robustly in its training environment, it experiences a moderate performance drop in a novel environment. This suggests that although the agent generalizes reasonably well, some aspects of its learned behavior are tuned specifically to the training conditions, leading to reduced effectiveness when those conditions change. **We refer to the source code for examples: for instance, a Sarsa agent went from an average score of 22 on the original "env" framework to a score of 15.5, i.e a drop of around 30% in performance**, when placed in a modified environment (which was larger, so that new (dx, dy) distances were unseen in the policy)

5 Conclusion - Presentation of Results

Our results, presented through state-value function plots, learning curves, and parameter-sweep graphs, reveal distinct performance patterns for the two agents. The Monte Carlo agent, while conceptually straightforward, exhibits higher variance and poorer abilities compared to the SARSA(λ) agent. Despite these advantages, both agents show a noticeable degradation in performance when transitioning to a new environment configuration, underscoring the challenges of policy generalization. Overall, these findings contribute valuable insights into the trade-offs between episodic and stepwise learning approaches in reinforcement learning.

References

1. A. Boistard [Online]. Source code available: https://github.com/alex-b106/Reinforcement-Learning/tree/master/Assignment
2. S. Christodoulidis. *Text-Flappy-Bird-Gym*. [Online]. Available: https://gitlab-research.centralesupelec.fr/stergios.christodoulidis/text-flappy-bird-gym
3. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.
4. Talendar. *Flappy-Bird-Gym*. [Online]. Available: https://github.com/Talendar/flappy-bird-gym