Comparative Analysis of Q-learning and SARSA in the Flappy Bird Game

1st Kha Duy Phan
Université Côte d'Azur, France
duy-kha.phan@etu.univ-cotedazur.fr

2nd Huy Dinh Nguyen
Université Côte d'Azur, France
dinh-huy.nguyen@etu.univ-cotedazur.fr

Abstract—**This report analyzes the performance of Q-learning and SARSA in training an agent for the Flappy Bird game. An enhanced reward function is designed to encourage prolonged flight, optimize gift collection, reward strategic actions, and penalize collisions. The study covers game mechanics, state representation, action dynamics, reward structure, learning algorithms, and experimental evaluation, with insights on performance, gift acquisition efficiency, and potential future improvements.**

Index Terms—**Reinforcement Learning, Q-learning, SARSA, Flappy Bird, Reward Functions.**

I. Introduction

Reinforcement Learning (RL) is extensively used to train agents for sequential decision-making problems. In Flappy Bird, a side-scrolling game, the agent controls a bird that moves forward while being influenced by gravity. The only available action is flapping, which briefly increases the bird's upward velocity. The primary goal is to maneuver through dynamically placed pipe gaps while avoiding collisions to maximize both survival time and score.

Beyond obstacle avoidance, the agent can also collect gifts that spawn at different locations along its flight path. These gifts offer bonus rewards, incentivizing exploration of alternative flight strategies. However, acquiring them introduces a trade-off between risk and reward, as adjusting the trajectory to obtain gifts may increase the likelihood of crashes.

II. Game Description and Mechanics

A. Game Elements and Dynamics

- **Visual Elements:** The game features a background, moving ground, pipes as obstacles, and a bird controlled by the agent. A bonus element, referred to as a "gift", is also included.
- **Dynamics:**
  - The bird is affected by gravity and can flap to gain altitude.
  - Pipes continuously move leftwards at a constant speed, with new pipes generated as the older ones exit the screen.
  - The gift moves along with the pipes; when collected, it provides an extra reward.

B. State Representation and Actions

The agent's perception of the environment is defined by its state representation and the available actions.

1) State Representation: The positions of the bird, pipes, and gifts are represented by their respective (x,y) coordinates. Notably, the bird's x-coordinate remains constant. The state of the game is represented as a tuple capturing essential information related to the bird's position and the environment:

- **X-coordinate of the next pipe**
- **Vertical distance to the next pipe:** The difference in height between the bird and the lower edge of the upper pipe
- **X-coordinate of the next gift**
- **Vertical distance to the next gift:** The difference in height between the bird and the gift

Due to the continuous nature of the game, representing each state at the pixel level would result in an excessively large state space, making learning inefficient. To address this, the game environment is divided into discrete areas, where each area groups multiple pixel positions together. The agent considers all states within the same area as equivalent and applies the same action policy.

This discretization reduces complexity, accelerates learning, and enables more effective generalization across similar game scenarios. By mapping the bird's position, pipes, and gifts to predefined grid-like areas, the agent can make consistent decisions without the need for an impractically large state space.

2) Actions: Action selection is implemented using greedy strategy. Accordingly, the agent has two possible actions:

- **Action 0 (do nothing):** The bird is allowed to descend naturally under gravity.
- **Action 1 (flap):** The bird flaps its wings to gain altitude, counteracting gravity.

C. Enhanced Reward Function

The reward function is designed to provide clear, multidimensional feedback to guide the agent's learning:

- **+200:** Awarded for successfully passing through a pipe.
- **+500:** Granted for collecting a gift.
- **-50:** Penalizes the agent for missing a gift.
- **-1000:** Imposed when the bird collides with a pipe or the ground, leading to the termination of the episode.

This reward structure balances positive reinforcement for desirable actions with penalties for mistakes, thereby shaping the agent's behavior effectively.

III. Methodology

Both Q-learning and SARSA are implemented within the same framework for state discretization and Q-table management. Their primary difference lies in the update mechanism:

- **Q-learning:** An off-policy method that updates Q-values using the maximum expected future reward from the next state. This approach typically converges faster, though it may promote riskier strategies.
- **SARSA:** An on-policy method that updates Q-values based on the actual action taken in the next state, leading to a more conservative and stable learning process.

IV. Experimental Evaluation

A. Setup

The training process is configured with the following parameters: learning rate (α): 0.1, discount factor (γ): 0.9, number of training episodes: 800.

B. Results and Discussion

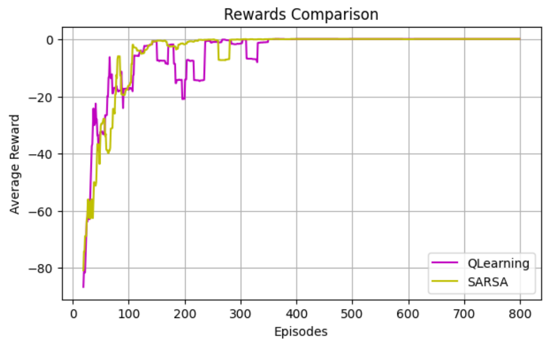

Fig. 1. Average rewards of Q-learning and SARSA

In Figure 1, the reward convergence of Q-Learning and SARSA follows a similar trend, with both algorithms experiencing rapid improvement in the early episodes. SARSA demonstrates a smoother learning curve, stabilizing slightly earlier than Q-Learning. In contrast, Q-Learning exhibits more fluctuations, indicating a greater degree of exploration. After approximately 300 episodes, both methods achieve near-optimal performance, with average rewards converging close to zero.

Figure 2 illustrates the mean score progression of Q-Learning and SARSA over 800 episodes. Q-Learning learns faster in the mid-phase (400-600 episodes), while SARSA stabilizes more gradually. In the final phase (700+ episodes), SARSA surpasses Q-Learning, achieving higher peak scores with less fluctuation. Furthermore, Figure 3 shows the gift collection efficiency of Q-Learning and SARSA over 800 episodes. SARSA learns to collect gifts earlier, but Q-learning catches up and achieves a similar capture rate (0.7 gifts per score) in later episodes. This indicates that SARSA learns to collect gifts earlier, but both can optimize the gift collection strategy in the long run.

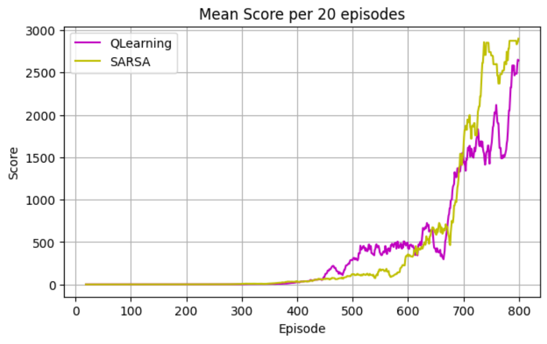

Fig. 2. Mean score of Q-learning and SARSA

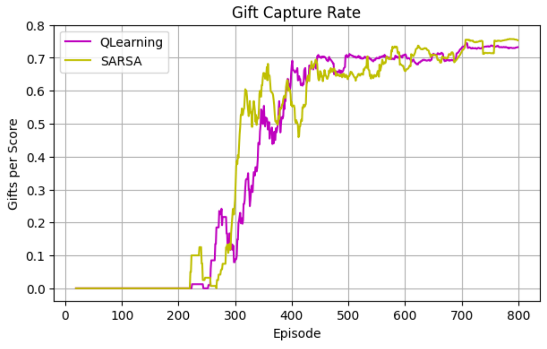

Fig. 3. Gift capture rate of Q-learning and SARSA

V. Discussion and Conclusion

A. Discussion

The results demonstrate that SARSA and Q-Learning exhibit distinct learning behaviors in training a Flappy Bird agent. SARSA shows more stable learning, achieving consistent rewards, higher peak scores, and earlier gift collection efficiency. In contrast, Q-learning learns faster initially but experiences greater fluctuations, reflecting its off-policy nature and tendency for broader exploration. Despite SARSA's early advantage in gift collection, Q-learning eventually optimizes its strategy, leading to a similar gift capture rate. In long run, both algorithms converge towards effective policies.

B. Limitations and Future Work

While the results are promising, several limitations remain:

- The discretization of continuous states might omit some dynamic details of the game.
- Further parameter tuning may improve performance.
- Future work could explore more complex reward structures and advanced state representation techniques, possibly incorporating deep learning methods.

C. Conclusion

The study compares Q-Learning and SARSA in training a Flappy Bird agent, highlighting their distinct learning behaviors. Besides, both algorithms demonstrate their effectiveness in this task given sufficient training conditions.