Applying Q-Learning to Rock, Paper, Scissors: A Reinforcement Learning Approach

Daizy Buluma and Gretta Ineza
COSC-241 (AI) Course Final Project Report, Amherst College

***Abstract*—This project explores the application of Q-learning, a reinforcement learning algorithm, to the classic Rock, Paper, and Scissors (RPS) game. The primary goal was to develop an AI agent capable of learning optimal strategies through interactions with both predefined and human opponents. The agent was evaluated against a random opponent, a patterned opponent, and human players. The results show that the Q-learning agent successfully adapted to both predefined and human strategies, demonstrating the effectiveness of reinforcement learning in dynamic environments. However, the absence of techniques such as epsilon decay and dynamic windowing presented limitations in performance optimization.**

***Index Terms*—Reinforcement Learning, Q-Learning, Rock Paper Scissors, AI, Human-Computer Interaction.**

I. Introduction

The game of Rock, Paper, Scissors (RPS) is a simple yet widely recognized game of chance. Despite its apparent simplicity, the game presents a compelling platform for exploring reinforcement learning techniques. By attempting to learn optimal strategies over time, an AI agent can transform this stochastic environment into an intriguing testbed for reinforcement learning algorithms.

A. Objective

This project implements Q-learning to develop an AI agent capable of playing RPS against both predefined opponents and human players. The agent learns by adapting its strategy based on rewards obtained from interactions, in order to maximize its long-term performance. Specific goals include:

- Implementing Q-learning to model decision-making processes.
- Evaluating the agent's ability to adapt to predefined patterns and human decision making.
- Demonstrating the practical application of reinforcement learning in a competitive real-time scenario.

B. Related Work

Q-learning has been widely applied in multi-agent systems to develop intelligent strategies for non-deterministic games. Previous works such as Mnih et al. (2015) and Silver et al. (2016) have demonstrated the efficacy of reinforcement learning in more complex games like Go and Chess. However, there is limited research specifically focused on applying Q-learning to the game of Rock, Paper, Scissors. This project seeks to fill that gap by exploring the use of Q-learning in a simple game setting and evaluating its performance against various opponents.

II. Motivation and Background

The motivation for applying Q-learning to RPS stems from its simplicity as a test case to explore core concepts in reinforcement learning. Unlike more complex games, RPS allows for rapid prototyping and focused evaluation of algorithmic learning behavior. While Q-learning has been applied to complex games like Go and Chess, its application in simple, real-time competitive settings like RPS remains under explored.

A. Research Questions

1) How effectively can Q-learning be applied to the game of Rock, Paper, Scissors?
2) How effectively can a Q-learning agent adapt its strategy against opponents with different levels of predictability (random, patterned, and human)?
3) What are the key performance trends in terms of win rate and average reward as the agent learns over time?
4) What limitations and potential improvements can enhance the Q-learning agent's performance in dynamic environments?

III. Methodology

A. Algorithm

The Q-learning algorithm was chosen for its ability to learn optimal policies without requiring a model of the environment. Q-learning is a model-free reinforcement learning algorithm that learns a policy to maximize the cumulative reward. At its core, the algorithm updates the Q-values using the Bellman equation. Key parameters included:

- **State Space:** The opponent's last move.
- **Action Space:** The three possible actions in RPS: rock, paper, and scissors.
- **Reward Scheme:** +1 for a win, 0 for a tie, and -1 for a loss.
- **Policy:** An epsilon-greedy policy for action selection, balancing exploration and exploitation.

B. Opponent Types

The agent was evaluated against three opponent types:

1) **Random Opponent:** Selects actions randomly.
2) **Patterned Opponent:** Follows a fixed sequence of moves.
3) **Human Opponents:** Two human players participated in the experiment, providing a dynamic and unpredictable environment.

C. Implementation

The Q-learning agent was implemented in Python. Key features of the implementation include:

- **Environment:** The agent interacts with an environment class managing state transitions and rewards.
- **Agent Design:** A QLearningAgent class handles action selection, learning updates, and Q-value management.
- **Opponents:** Random and patterned opponents simulate predefined strategies, while a Tkinter-based interface allows human interaction.
- **Visualization:** Real-time plotting using Matplotlib shows reward trends over episodes.

The full implementation is available in this Google Drive Folder: https://drive.google.com/drive/folders/1ONuBgLeiNuwsveELvqjwQ_eXn20kTM6a?usp=sharing.

IV. Evaluation Methodology

The performance of the Q-learning agent was measured using the following metrics:

1) **Win Rate:** The proportion of games won by the agent.
2) **Average Reward:** The mean reward accumulated over episodes.
3) **Learning Speed:** How quickly the agent's performance improved during training.

Experiments were conducted over:

- **10,000 episodes** against random and patterned opponents.
- **100 rounds** against each human player.

V. Results

A. Against Random Opponent

The agent achieved a win rate close to 33%, consistent with random play, indicating no exploitable patterns in the opponent. This addresses **Research Question 2** by showing that the agent cannot exploit randomness.

- **Win Rate Trend:** The graph indicates a stabilization at a win rate of 0.34, after initially showing small fluctuations.
- **Average Reward:** The average reward remains stable at 0.0, reflecting limited improvement in reward accumulation.

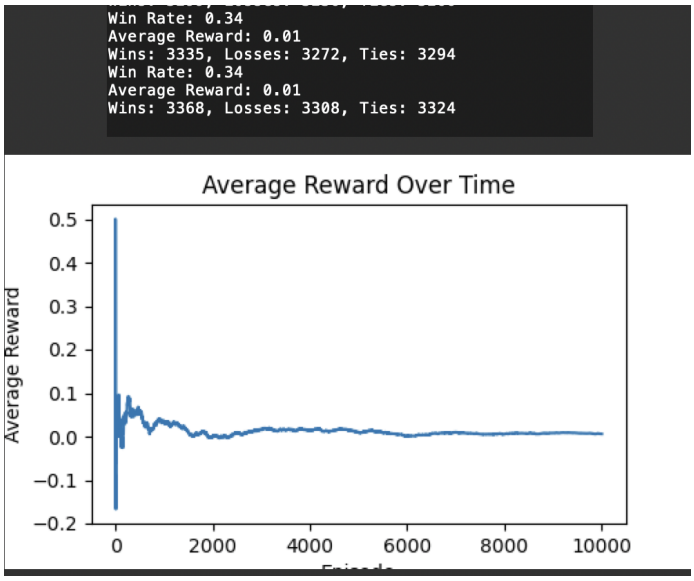

Fig. 1. Performance against a random opponent: Average Rewards rate over episodes.

B. Against Patterned Opponent

The agent's win rate increased significantly over time as it learned to exploit the opponent's predictable sequence.

- **Win Rate Trend:** Graphical data shows steady improvement over episodes, with a peak win rate of 80%.
- **Average Reward:** The average reward rate stabilizing at about 0.6 aligned with learning progress.

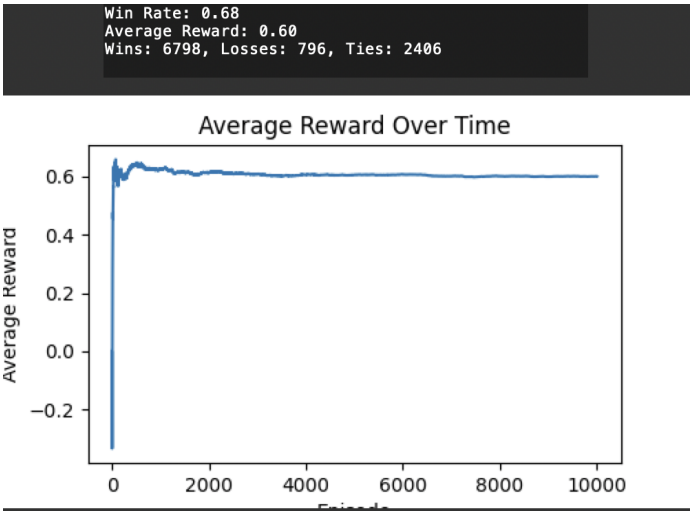

Fig. 2. Performance against patterned opponent: Average reward over time.

This result highlights the agent's ability to adapt to structured behavior, as posed in **Research Question 2** and partially in **Research Question 3**.

C. Against Human Opponents

The agent demonstrated adaptability by improving its win rate over successive rounds, showcasing its ability to learn and counter human decision-making tendencies.

- **Human Player 1:** The agent achieved a win rate of 63% over 100 rounds.
- **Human Player 2:** The agent achieved a win rate of 50% over 100 rounds, reflecting varied strategy patterns.

These findings reinforce the relevance of **Research Question 1** and the trends from **Research Question 2**.

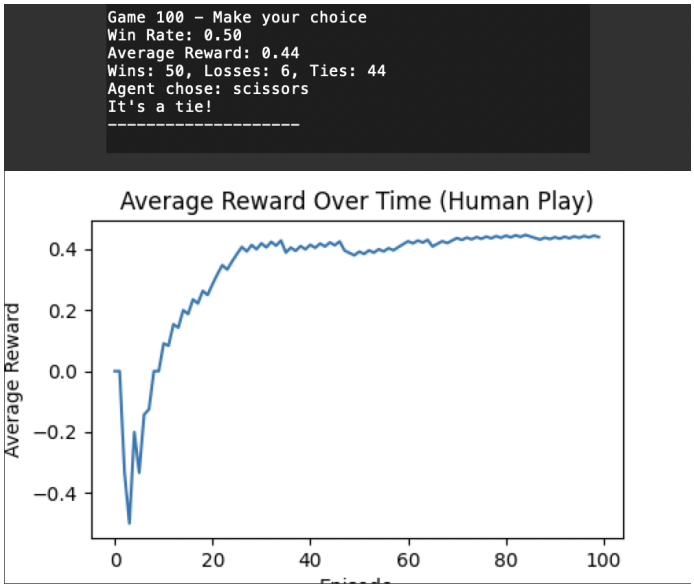

Fig. 3. Performance against human opponents: Average Reward over rounds.

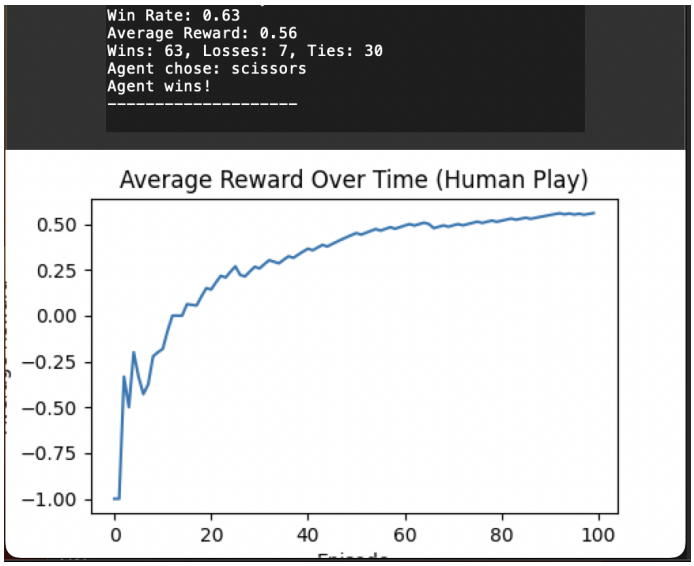

Fig. 4. Performance against human opponents: Average Reward over rounds.

VI. Conclusions

This project successfully demonstrated the application of Q-learning in a relatively simple competitive setting, showcasing the potential of reinforcement learning in dynamic environments. The agent was able to effectively learn strategies to compete against both predefined and human opponents, thus validating the utility of Q-learning as a method for developing intelligent decision-making systems. The agent's ability to adapt and improve its strategies over time highlights the power of reinforcement

learning in scenarios where trial and error play a key role in learning optimal behaviors.

However, despite these successes, certain limitations in the implementation restricted the agent's overall performance. Notably, the absence of epsilon decay and dynamic windowing mechanisms hindered the agent's exploration-exploitation balance and its ability to effectively adjust to changing opponent strategies. These shortcomings prevented the agent from reaching its full potential in terms of both adaptability and long-term performance.

To address these limitations, future work could incorporate advanced techniques to enhance the agent's learning capabilities:

- **Deep Q-learning:** By using deep neural networks to approximate the Q-function, the agent could handle more complex environments and larger state spaces, enabling it to learn better strategies in more challenging scenarios.
- **Epsilon Decay:** Implementing epsilon decay would allow the agent to start with a high exploration rate and gradually shift towards exploitation as it gains more experience, thus improving the efficiency of the learning process.
- **Dynamic State Representation:** Enhancing the state representation to capture more intricate opponent behaviors and environmental dynamics could lead to more robust and flexible strategies, improving performance in more complex competitive settings.

Overall, this project underscores the promise of reinforcement learning, particularly Q-learning, in simple yet challenging environments such as Rock-Paper-Scissors (RPS). It demonstrates that while reinforcement learning can be highly effective in these settings, there is still much room for improvement. The insights gained from this project not only provide valuable answers to questions about Q-learning's effectiveness but also set the stage for future enhancements that could unlock even greater potential in real-world applications.

REFERENCES

[1] V. Mnih, et al., "Human-level control through deep reinforcement learning, 2015.

[2] D. Silver, et al., "Mastering the game of Go with deep neural networks and tree search, 2016.