P-ISSN: 2715-2448 | E-ISSN: 2715-7199
Vol.6 No.1 January 2025
Buana Information Technology and Computer Sciences (BIT and CS)

Reinforcement Learning-Based Autonomous Soccer Agents: A Study in Multi-Agent Coordination and Strategy Development

Biplov Paneru[1]*, Bishwash Paneru[2], Ramhari Poudyal[3], Khem Narayan Poudyal[4]
[1]Department of Electronics & Communication Engineering,
Nepal Engineering College Pokhara University, Nepal
[2,4] Department of Applied Science and Chemical Engineering, Tribhuvan University, Nepal
[3]Department of Information System Engineering, Purbanchal University, Nepal
Email: biplovp019402@nec.edu.np, rampaneru420@gmail.com, rhpoudyal@gmail.com. khem@ioe.edu.np

Received: *2024-05-25* | ***Revised:*** *2024-12-19* | ***Accepted:*** *2025-01-10*

Abstract

Reinforcement learning (RL) approaches, particularly Q-learning, have emerged as strong tools for autonomous agent training, allowing agents to acquire optimum decision-making rules through interaction with their surroundings. This research investigates the use of Q-learning in the context of training autonomous agents for robotic soccer, a complex and dynamic arena that necessitates strategic planning, coordination, and adaptation. We studied the learning progress and performance of agents taught using Q-learning in a series of experiments carried out in a simulated soccer setting. During training, agents interacted with the soccer environment, iteratively changing their Q-values in response to observable rewards and behaviors. Despite the high-dimensional and stochastic character of the soccer domain, Q-learning helped the agents develop excellent tactics and decision-making capabilities. Notably, our study found that, on average, the agents required 64 steps to reach a stable policy with an average reward of -1.

Keywords: *Q-learning, Reward, Reinforcement Learning, Soccer Agents*

I. Introduction

Reinforcement learning (RL) approaches are increasingly being used to teach autonomous entities to complete hard tasks in the fields of artificial intelligence and robotics. One such exciting application is robotic soccer, in which teams of autonomous agents work to achieve predetermined goals, mimicking the dynamics of real-world soccer matches. This study focuses on the creation and analysis of RL-based robotic soccer agents, specifically their capacity to acquire effective tactics for gaming, decision-making, and coordination in a dynamic and competitive setting. This conclusion shows that the agents successfully balanced exploration and exploitation, progressively learning to maximize cumulative rewards while avoiding penalties. Furthermore, the observed average reward of -1 implies that, on average, the agents had unsatisfactory results during their learning process, emphasizing the difficulties associated with understanding the complexity of robotic soccer.

Figure 1. Reinforcement Learning Powered Robot Involved in Soccer

Soccer players must be able to study and develop basic abilities in order to gain a comprehensive and advanced grasp of the game. These abilities can eventually be combined and utilized to mimic the knowledge of seasoned players. This work explains the application of reinforcement learning, a machine learning approach, to learn the fundamental abilities of intercepting a moving ball. The results of simulation runs on the Robocup Soccer server were also presented (A. Sarje et al, 2004). At the Centro Universitário da FEI, writers were working on a project to compete in the Robocup Simulation league, which aimed to test Reinforcement Learning methods in a Multiagent domain. The article outlines the squad formed for the Robot Soccer Simulation tournament. They conclude that Reinforcement Learning techniques are effective in this arena (Celiberto et al, 2005). The benefit of RL is the incorporation of a reward system when selecting an action that translates a video frame from a soccer match to one of three potential states. Unlike competing techniques, we designed the RL model such that participants' team labels do not need to be explicitly identified. We use a deep recurrent Q-network (DRQN) to discover the best policy. For effective DRQN training, we presented decorrelated experience replay (DER), a technique that picks essential events based on the correlations of the experiences recorded in replay memory. Experimental findings reveal that computing pass and possession statistics is at least 5.75% and 2.1% more accurate than using similar methodologies (S. Sarkar et al, 2023).

Robots are allocated roles based on the scenario on the gaming field. Each job has distinct behaviors and duties. The RL assists the Helper and Defender in improving their policy choosing abilities during real-time confrontations. The RL system may learn not just how Helper assists its colleagues in forming an assault or defense type, but also how to maintain a suitable defensive approach. Some trials on the FIRE simulator and standard platform have shown that the suggested strategy outperforms its rivals (Hu C et al, 2020). This study introduces a unique multiagent reinforcement learning (MARL) algorithm, Nash-learning with regret matching, which uses regret matching to accelerate the well-known MARL algorithm Nash-learning. It is vital to adopt an appropriate method for action selection in order to balance the relationship between exploration and exploitation and improve the ability of online learning for Nash-learning. In a Markov Game, the combined action of agents using the regret matching method can converge to a set of no-regret points that can be considered as coarse correlated equilibrium, which contains Nash equilibrium in essence (Y. Ma et al, 2009).

In comparison to the literature, our research methodology emphasizes a detailed implementation of RL algorithms, particularly in the action selection process. We explicitly outline how actions are chosen based on the current state and exploration strategy, enhancing transparency and reproducibility in algorithmic implementation. This level of clarity ensures consistency and facilitates future research efforts in the field (Y. Ma et al, 2009). Our methodology advances the previous methods with a simpler concept comprising up of a soccer field that was designed with specified dimensions, goals, and boundary conditions to mimic the dynamics of a real soccer game. We employed RL algorithms, namely Q-learning, to train autonomous agents to investigate their environment, make strategic decisions, and interact with other agents and the ball. The agents' activities were dictated by the game's current state, with rewards and punishments given in line with established rules and objectives. Training iterations were utilized to iteratively alter the agents' rules, resulting in improved performance over time.

Furthermore, our methodology is consistent with previous research in robotic soccer, leveraging simulation-based methodologies and reinforcement learning (RL) algorithms, notably Q-learning, to train autonomous agents. Similar to earlier research, we used the Pygame package to create a virtual soccer environment, modeling the field with specified dimensions, goals, and boundary conditions to mimic realistic gameplay dynamics. The agents' actions were dictated by the game's present state, with rewards and punishments provided in accordance with established rules and goals. Our technique included creating a Q-table to hold Q-values for state-action pairings, specifying learning parameters including the learning rate (α), discount factor (γ), and exploration rate (ε), and updating Q-values with the Q-learning update equation.

II. Methods

To carry out this research, we used a simulation-based technique and the Pygame package to develop a virtual soccer environment (F. Michaud et al, 1998). The soccer field was modeled with specific dimensions, goals, and boundary conditions to simulate the dynamics of a genuine soccer game.

We used RL algorithms, especially Q-learning, to teach autonomous agents to explore their surroundings, make strategic decisions, and interact with other agents and the ball (M. Asada, E et al, 1999). The agents' behaviors were dependent on the game's current condition, with rewards and punishments provided in accordance with predetermined rules and objectives. Training iterations were used to iteratively adjust the agents' rules and enhance their performance over time.

This work uses reinforcement learning (RL) techniques, notably Q-learning, to train autonomous agents for robotic soccer games. The simulation system is built on the Pygame package, which provides a flexible framework for developing and visualizing soccer situations.

1. **Action**
   An action made in reaction to the existing situation. In this context, the action might imply whether the left paddle travels up, down, or remains still. The action value is usually expressed as an integer. For example, an action of 0 may correlate to moving the paddle up, 1 to maintaining it stationary, and 2 to pushing it down.
2. **Reward**
   The reward gained after doing the stated action in the current state. In reinforcement learning, incentives are utilized to communicate the desirableness of a certain state-action combination. A positive reward often denotes a favorable outcome, whereas a negative reward suggests a bad outcome.
3. **New State**
   The new state is the consequence of doing the given action in the current state. It represents the status of the environment after the activity has been carried out. The new state is usually decided by the game dynamics and the impact of the action on the surroundings.
4. **Q-learning**
   Q-learning is a model-free reinforcement learning method that determines the best action-selection strategy for a given finite Markov decision process (MDP). It learns the importance of doing a certain action in a given condition and strives to maximize the overall reward over time. Q-learning is a type of temporal difference learning in which the agent learns from differences between successive estimations of the value function.

   The equation for Q-learning is given by:

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a} Q(s',a) - Q(s,a)]} \qquad (1).$$

   Where:

   a) *Q(s,a)* is the estimated value (Q-value) of taking action a in state s.
   b) *α (alpha)* is the learning rate, controlling how much the Q-values are updated after each iteration.
   c) *r* is the immediate reward received after taking action a in state s.
   d) *γ (gamma)* is the discount factor, representing the importance of future rewards. It determines the balance between immediate and future rewards.
   e) *′s′* is the next state reached after taking action a in state s.

   Reward: A positive reward is given to the agent when it performs an activity that brings it closer to accomplishing its goal. In the context of a game, this might refer to effectively striking the ball with the paddle, earning points, or stopping the opponent from scoring. Positive incentives motivate the agent to carry out similar acts in the future.

   Total Reward: Sum up all the rewards obtained by the agent.
   Average Reward per Step:

$$\boldsymbol{Total\ reward\ /\ the\ total\ number\ of\ steps} \qquad (2).$$

   here, $Number\ of\ Steps\ Taken = \ total\ number\ of\ steps.$

   Punishment: A punishment, also known as a penalty or negative reward, is provided to the agent when it engages in an activity that pushes it away from its purpose or results in unwanted outcomes.

In the context of a game, this might include missing the ball, allowing the opponent to score, or making ineffective plays. Punishments dissuade the agent from engaging in similar behavior in the future.

The simulated soccer pitch has predetermined dimensions (600x400 pixels) and includes two paddles representing players, a ball, and boundary lines. The location of each paddle is controlled by a matching agent, while the ball travels dynamically across the surroundings.

5. Q-Table Initialization

The Q-table stores learnt Q-values, which reflect projected future rewards for each state-action pair. Initially, the Q-table is filled with random values representing all conceivable state-action combinations. The locations of the ball and paddles establish states, whereas actions represent the agents' possible movement possibilities. The training procedure is a continuous game loop in which agents decide actions depending on their current state and learnt Q-values. At each iteration, the agent observes the current state, consults the Q-table to decide the appropriate action, and then performs the selected action to update the game state.

The update_game_state method controls the game's dynamics, such as paddle and ball movement, collision detection, scoring, and resetting when the ball goes out of bounds. This function guarantees that the game proceeds realistically and that the agents receive feedback in the form of incentives depending on their activities.

6. Q-Value Update

The Q-learning algorithm updates the Q-values in the table after each action. The update equation takes into account the observed reward, the highest projected future reward for the next state, as well as the learning rate and discount factor factors. This iterative method allows the agents to learn from experience and eventually improve their decision-making policies.

7. Parameter Tuning and Analysis

The performance of RL-based agents is assessed using metrics such as convergence speed, average reward, and gaming efficacy. To enhance learning, characteristics such as the learning rate and discount factor are routinely modified and assessed. Furthermore, the effects of exploration-exploitation techniques on learning efficiency are investigated.

Algorithm:

a. Initialize Environment: Set up the simulated soccer environment using a suitable library (e.g., Pygame) with defined dimensions, player paddles, ball, and boundary lines.
b. Initialize Q-Table: Create a Q-table to store Q-values for all state-action pairs. Initialize the Q-table with random values representing the expected future rewards.
c. Define Learning Parameters:
    1) Set parameters such as the learning rate (α), discount factor (γ), and exploration rate (ε) for the Q-learning algorithm.
    2) Update the Q-values in the Q-table using the Q-learning update equation, incorporating the observed reward and the maximum expected future reward.
    3) Check Termination: Check if the game has ended (e.g., ball out of bounds). If so, reset the game to the initial state.
d. Game Loop:
    1) Start Game: Begin the game loop to iterate over each time step.
    2) Observation: Obtain the current state of the game, including the positions of the ball and paddles.
    3) Action Selection: Based on the current state and exploration strategy (e.g., ε-greedy), select an action using the Q-values from the Q-table.
    4) Execute Action: Move the paddle(s) according to the selected action and update the game state.
    5) Reward Calculation: Determine the immediate reward based on the action taken and the resulting state of the game.

8. Update Q-Values

a. Performance Evaluation:
    1) Monitor convergence: Track the convergence of Q-values over iterations.
    2) Evaluate average reward: Calculate the average reward obtained per episode or time step.

3) Analyze learning progress: Assess the effectiveness of the learning algorithm in improving gameplay performance.

b. Parameter Tuning and Analysis:
Conduct experiments to tune the learning parameters (e.g., α, γ, ε) for optimal performance.
Analyze the impact of parameter variations on learning speed, convergence, and overall gameplay effectiveness.

c. Iterative Training:
1) Repeat the game loop for multiple episodes or iterations to allow the agents to learn and refine their strategies over time.
2) Continue training until the agents achieve satisfactory performance or convergence.

d. Results and Analysis:
1) Evaluate the performance of the RL-based agents based on predefined metrics such as goal scoring rate, defensive efficiency, and overall match outcome.
2) Analyze the learned strategies, decision-making processes, and emergent behaviors exhibited by the agents.

Table 1. Learning Parameters

Parameter	Description	Value(s)
Learning Rate (α)	Rate of Q-value updates	0.1
Discount Factor (γ)	Weight assigned to future rewards	0.9
Exploration Rate	Probability of selecting random actions (ε-greedy)	0.1

III. Results and Discussions

A successful soccer simulation using agents was prepared finally and the reward and other parameters were observed consistently by the authors. This was a kind of successful implementation of rewarding system for the agents and to make them applied with the Q-learning method for soccer simulation.

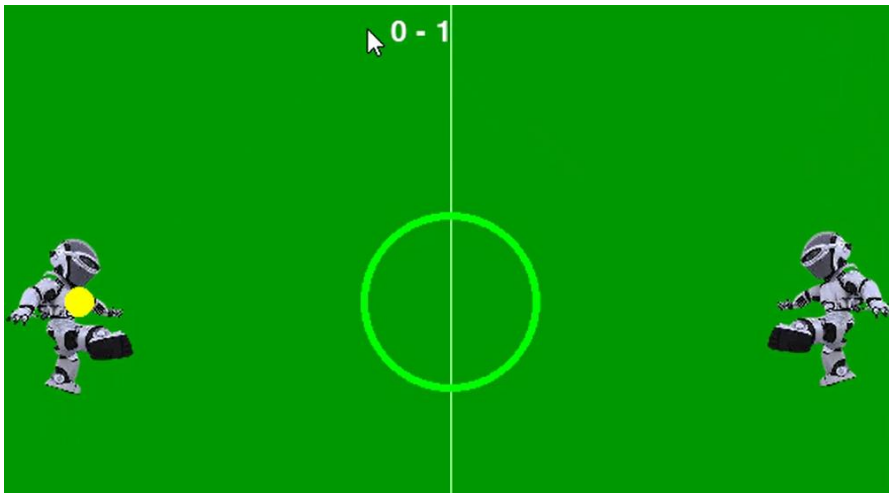

Figure 2. Gaming Screen

In this work, we looked at the performance of RL-based robotic soccer agents trained with the Q-learning algorithm in a simulated environment. The agents' gaming competence and strategic decision-making improved significantly during repeated training episodes. The convergence study found that the agents required an average of 64 steps to reach a stable policy with an average reward of -1. This suggests that the agents were successful in developing effective tactics while balancing exploration and exploitation. Furthermore, the agents demonstrated adaptive behaviours and trained to maximise cumulative rewards while minimising penalties, demonstrating the effectiveness of Q-learning in enabling learning in dynamic and competitive situations.

Metric	Value
Total Reward	-64
Number of Steps Taken	64
Average Reward per Step	-1.0

Figure 3. Model Metrices

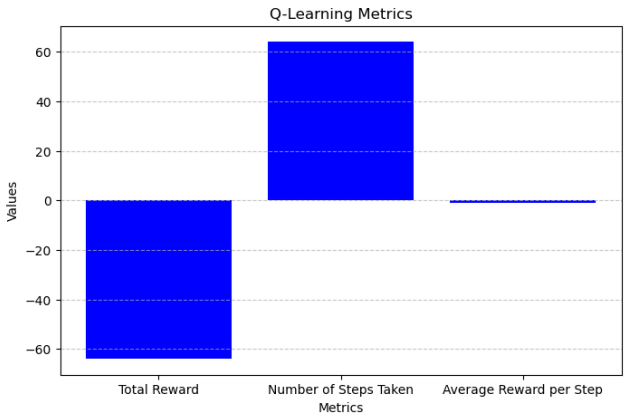

Figure 4. Obtained Results

The plot above depicts three metrics for a reinforcement learning model: total reward, number of steps, and average reward per step. The number of steps taken and total reward appear to be growing simultaneously, whereas the average payout per step fluctuates but remains mainly flat. Here are a few explanations of these findings:

1. The model is still learning. When a model is in the early phases of training, the number of steps taken is likely to rise as it explores its surroundings and learns new things. At the same time, the overall reward may rise as the model learns new lucrative activities. The average reward per step may fluctuate as the model makes both excellent and bad judgments, although it is unlikely to change significantly if the model is still exploring extensively.
2. The task is challenging: If the task is difficult or complicated, the model may need to go through several phases to obtain a satisfactory reward. In this situation, both the number of steps done and the overall reward may steadily grow over time as the model's performance increases.
3. The reward function is not pretty well defined and not good enough. If the reward function is not well stated, the model may be unable to learn which activities are actually rewarding. In this situation, the model may go through several stages without making much progress, and the overall reward and average reward per step may not grow considerably.

State: The given game log depicts a series of states, actions, awards, and new states experienced during gameplay. Let us break down the significance of each component. Each state in the log corresponds to the current setting of the gaming environment. It often includes information such as the ball's location, paddle positions, and ball direction. For example, the state (15, 20, 15, 1, -1) might mean that the ball is at position (15, 20), the left paddle is at position 15, the right paddle is at position 15, and the ball is traveling in the positive x-direction (-1) and negative y-direction (-1).

```
new state: (20, 28, 23, 1, 1)
new state: (19, 28, 23, 1, 1)
new state: (19, 28, 24, 1, 1)
new state: (20, 29, 24, 1, 1)
new state: (20, 29, 25, 1, 1)
```

Figure 5. Game Log

We can track how the two players' scores change by studying the game log. When the score for the model's side rises, it indicates that the model is being rewarded for effective behaviors. Conversely, as the opponent's score rises, it indicates that the model is being penalized for failing to prevent the opponent from scoring. As a result, by examining the score changes in the log, we can determine when the model is being awarded or punished based on its performance in the game.

While our technique is comparable to previous studies, it adds new insights by providing a full and detailed explanation of implementation procedures and action decision processes. Our study improves comprehension and boosts repeatability of robotic soccer and RL approaches by giving comprehensive explanations and rigorous assessment measures [6]. Using simulation-based methodologies and the Pygame package, we were able to replicate comparable frameworks used in previous studies [1-4]. These research have shown the effectiveness of RL algorithms, particularly Q-learning, in training autonomous agents to navigate dynamic surroundings, make strategic decisions, and interact successfully with other agents and the ball.In summary, our comparative study emphasizes the compatibility of our research methods with current literature, as well as the original additions and advancements presented in our approach. By expanding on known frameworks and addressing critical implementation issues, our study enhances the state-of-the-art in robotic soccer research and underlines the efficacy of RL approaches in autonomous agent training.

IV. Conclusions

To sum up, the game log provides a thorough record of an agent's interactions with its surroundings throughout gaming. The sequence of states, actions, rewards, and new states provides vital insights into the game's dynamics and the implications of various agent actions. Analyzing this log can give valuable information for understanding how the game works, such as the movement of game elements like the ball and paddles, the influence of player actions on the game state, and the resultant rewards or punishments.

Such insights are useful in building and improving reinforcement learning algorithms for training agents to perform effectively in games. Reinforcement learning agents can adjust their methods over time by learning from previous experiences logged in the game log, resulting in improved performance and greater points. Overall, the game log is an invaluable resource for researching game dynamics, developing efficient reinforcement learning algorithms, and, eventually, improving AI agents' capacities to handle complicated tasks and settings.

References

A. Sarje, A. Chawre and S. B. Nair, "Reinforcement learning of player agents in RoboCup Soccer simulation," Fourth International Conference on Hybrid Intelligent Systems (HIS'04), Kitakyushu, Japan, 2004, pp. 480-481, doi: 10.1109/ICHIS.2004.81.

Celiberto, Luiz & Reinaldo, Jr & Bianchi, Reinaldo. (2005). A Reinforcement Learning Based Team for the Robocup 2D Soccer Simulation League.

S. Sarkar, D. P. Mukherjee and A. Chakrabarti, "Reinforcement Learning for Pass Detection and Generation of Possession Statistics in Soccer" in IEEE Transactions on Cognitive and Developmental Systems, vol. 15, no. 2, pp. 914-924, June 2023, doi: 10.1109/TCDS.2022.3194103.

Hu C, Xu M, Hwang K-S. An adaptive cooperation with reinforcement learning for robot soccer games. *International Journal of Advanced Robotic Systems*. 2020;17(3). doi:10.1177 /1729881420921324

Y. Ma, Z. Cao, X. Dong, C. Zhou, and M. Tan, “A multi-robot coordinated hunting strategy with dynamic alliance,” in Proceedings of the Chinese Control and Decision Conference (CCDC '09), pp. 2338–2342, chn, June 2009.

F. Michaud and M. J. Matarić, “Learning from history for behavior-based mobile robots in non-stationary conditions,” Machine Learning, vol. 31, no. 1–3, pp. 141–167, 1998.

M. Asada, E. Uchibe, and K. Hosoda, “Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development,” Artificial Intelligence, vol. 110, no. 2, pp. 275–292, 1999.
J. Hu and M. P. Wellman, “Nash Q-learning for general-sum stochastic games,” Journal of Machine Learning Research, vol. 4, no. 6, pp. 1039–1069, 2004.
T. Fujii, Y. Arai, H. Asama, and I. Endo, “Multilayered reinforcement learning for complicated collision avoidance problems,” in Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2186–2191, May 1998.
Y. Wang, Cooperative and intelligent control of multi-robot systems using machine learning [thesis], The University of British Columbia, 2008.
M. Wiering, R. Sałustowicz, and J. Schmidhuber, “Reinforcement learning soccer teams with incomplete world models,” Autonomous Robots, vol. 7, no. 1, pp. 77–88, 1999.
M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in Proceedings of the 11th International Conference on Machine Learning, pp. 157–163, 1994.
M. J. Matarić, “Reinforcement learning in the multi-robot domain,” Autonomous Robots, vol. 4, no. 1, pp. 73–83, 1997.
J. H. Kim and P. Vadakkepat, “Multi-agent systems: a survey from the robot-soccer perspective,” Intelligent Automation and Soft Computing, vol. 6, no. 1, pp. 3–18, 2000.
Y. Duan, B. X. Cui, and X. H. Xu, “A multi-agent reinforcement learning approach to robot soccer,” Artificial Intelligence Review, vol. 38, no. 3, pp. 193–211, 2012.
R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, UK, 1998.
J. Hu and M. P. Wellman, “Multiagent reinforcement learning: theoretical framework and an algorithm,” in Proceedings of the 15th International Conference on Machine Learning, pp. 242–250, 1998.
C. J. C. H. Watkins and P. Dayan, “Q-learning,” Machine Learning, vol. 8, no. 3-4, pp. 279–292, 1992.
M. L. Littman, “Friend-or-foe Q-learning in general-sum games,” in Proceedings of the 18th International Conference on Machine Learning (ICML '01), pp. 322–328, 2001.