Maxim Loukine
101275222
Project Proposal
Group 122
COMP3106A

Project Proposal: Reinforcement Learning Rocket League Bot

Background and Objectives

Rocket League is a physics-based soccer video game where players use rocket-powered cars to hit a ball into the opponent's goal. Its dynamic and continuous environment makes it an exciting testbed for artificial intelligence, especially reinforcement learning.

The focus of this project is to design and implement an AI-controlled Rocket League bot using reinforcement learning to learn how to play. A Q-learning algorithm that will gradually learn the expected long-term reward for each state-action pair in the game environment will be used. With repeated play, the agent will learn to approach the ball, make contact, and attempt to score goals against the built-in Rocket League bots. The use of an in-class algorithm connects the theoretical ideas of reinforcement learning with an applied, interactive gaming environment.

The objectives of the project are to implement a Q-learning agent with the capability of self-improvement through experience, evaluate its performance against that of baseline scripted bots, and analyze how its policy evolves over time. The final outcome will demonstrate the potential of reinforcement learning in complex real-time simulations.

Proposed Artificial Intelligence Methods
This project will apply the Q-learning algorithm to train the Rocket League bot. In Q-learning, the agent learns a value function Q(s,a)Q(s, a)Q(s,a), which estimates the expected cumulative reward of taking action aaa in state sss and following the learned policy thereafter. This is done by iteratively updating the Q-values with the Bellman update rule.

The state representation will include simple information about the position, velocity, distance, and angle of the car and ball. The agent's policy will be an ε-greedy exploration strategy: it selects random actions with probability ε and otherwise takes the best-known Q-value. This ε will decay over time to encourage exploitation of learned behaviors.

A reward function is used that gives positive feedback for desired events and negative feedback for mistakes. Large rewards are given when the agent scores a goal, smaller rewards are given after touching the ball or moving towards it, and own goals, inactivity, or moving away from the ball are penalized.

Dataset or Environment

Maxim Loukine
101275222
Project Proposal
Group 122
COMP3106A

No pre-existing data set will be necessary for this project, as all the data will be inherently created by the environment through interaction. Training will be done using the RLBot framework or the RocketSim API Python API to programmatically control Rocket League bots.

Validation and Analysis Strategy

The performance of the trained bot is going to be verified in repeated matches against default scripted bots. Quantitative metrics that will be measured include the win rate, average number of goals scored per match, and cumulative reward per episode. Convergence of Q-values and overall stability of the policy will also be analyzed. Qualitative evaluation will be performed by observation of gameplay videos to establish whether the bot acts in purposeful ways. Further experiments could compare different learning rates, discount factors, or exploration rates to understand their impact on learning speed and stability.

Novelty of the Project

Most of the Rocket League AI projects so far are based on scripted rules or deep neural networks; this project tries to implement classical Q-learning from scratch. This is novel because of mostly using a discrete reinforcement learning approach and applies it in a continuous, physics-driven environment. This work will demonstrate the process of taking these theoretical algorithms and adapting them to scale for big state and action spaces. Additionally, it gives insight into challenges of discretization and reward design in real-time environments, and both strengths and limitations of Q-learning.

Weekly Milestone Schedule

Following proposal approval, the project will proceed according to the following weekly plan.

Week 1: Set up the environment and test communication with Rocket League via RLBot or RocketSim.

Week 2: Implement the Q-learning framework, including Q-table initialization and ε-greedy policy.

Week 3: Design and test reward functions using short training simulations.

Week 4: Run full training loops, collect logs, and monitor cumulative reward trends.

Week 5: Evaluate performance through matches against built-in bots and adjust learning parameters.

Week 6: Optimize training and visualize results through plots of win rate, goals, and reward convergence.

Week 7: Compile final analysis, generate gameplay recordings, and prepare the report and presentation.