Investigation of Reinforcement Learning Methods for Raceline Optimization

Joseph Hadidjojo, Tung Pham

November 2, 2024

1 Introduction

Formula 1 (F1) racing is renowned for the intense precision, skill, and strategy that its drivers bring to the track. In each race, drivers must determine the fastest path around the circuit, or the "racing line," which enables them to maintain optimal speed while negotiating corners and straights. This process is not merely a physical feat; it requires an in-depth understanding of the vehicle's capabilities and a sharp sense of timing. Talented drivers often master the racing line in only a few laps of practice, fine-tuning it to suit their unique driving style and the distinct characteristics of their machinery.

Inspired by this mastery, the objective of this project is to explore the potential of reinforcement learning (RL) to approximate an optimal racing line on a simplified F1 track. The core idea is to use RL, where an agent learns through experience, to find the fastest path around a track by optimizing two key actions: acceleration and rotation. By framing the problem in this way, we aim to develop a model that could eventually serve as a foundational tool for amateur racers with limited experience.

However, this problem presents unique challenges compared to more conventional RL problems:

- **High-dimensional, continuous state and action spaces**: Unlike simpler tasks with discrete or limited action spaces, racing involves continuous control of speed, position, and rotation, all of which must be adjusted dynamically to adapt to the track layout.
- **Balancing competing objectives**: Achieving the optimal racing line requires the agent to balance maximizing speed with avoiding crashes or leaving the track boundaries. High speeds, especially around tight corners, increase the risk of the agent veering off course, demanding precise control over acceleration and rotation to maintain the ideal path within the track's limits.
- **Delayed rewards**: The consequences of an action, like setting up for a corner, may not be immediately evident. The agent must learn to associate

actions taken several turns ago with the eventual outcome, making the learning process more complex.

- **Track variability and precision demands**: Unlike simpler environments, the race track's varying curvature and the need for fine-grained control make it harder for the agent to generalize its learned strategy across different segments of the track.

These factors contribute to making the problem of finding an optimal race line particularly challenging for RL, setting it apart from simpler RL applications.

While F1 and professional racing teams likely already utilize advanced tools and systems to estimate the optimal racing line, these applications are typically proprietary and remain restricted to private use. Due to the competitive nature of the sport, insights and strategies derived from such systems are rarely disclosed to the public. Consequently, amateur racers and enthusiasts have limited access to data-driven methods for learning and improving their racing lines. By exploring this problem with reinforcement learning, we aim to create a foundation for a more accessible tool that can help individuals develop their skills and gain insights into racing line optimization without requiring the high costs or exclusivity of professional-grade systems. This project not only contributes to the field of RL in complex, continuous control tasks but also has the potential to democratize access to advanced racing analytics for a broader audience.

2 Background and Related Works

Reinforcement learning (RL) has seen significant advancements in recent years, especially in applications requiring continuous control and dynamic decision-making, such as autonomous driving and racing. In racing, RL agents must optimize not only for speed but also for safety, as they navigate complex tracks with tight turns and variable conditions. This section reviews notable works in RL and autonomous driving that have informed the development of racing agents, highlighting the techniques and challenges relevant to this project.

2.1 Deep Q-Network (DQN) for Autonomous Driving

A foundational work in deep reinforcement learning is the introduction of the Deep Q-Network (DQN) by Mnih et al. (2015) [5], which achieved human-level performance on Atari games. Although the DQN algorithm primarily focuses on discrete action spaces, it inspired adaptations for continuous and complex environments, such as autonomous driving and racing, where agents must navigate through tracks while balancing speed and control. This approach laid the groundwork for subsequent RL algorithms that could handle the continuous control demands of racing tasks.

2.2 Racing Agent with Reinforcement Learning

In racing, agents often benefit from continuous-action reinforcement learning algorithms, which allow finer control over movement variables such as speed and rotation. The Soft Actor-Critic (SAC) algorithm, introduced by Haarnoja et al. (2018) [2], has been applied in racing simulations for its ability to handle continuous action spaces with entropy regularization. SAC's exploration techniques are particularly effective for racing environments, where agents must strike a balance between fast exploration of new actions and reliable, safe driving.

2.3 Using RL for Track Navigation in Self-Driving Cars

Another significant contribution to continuous-action reinforcement learning is the Deep Deterministic Policy Gradient (DDPG) algorithm by Lillicrap et al. (2016) [3]. DDPG has shown promise in applications requiring high-precision control, such as car racing simulators, where agents must learn a delicate balance between acceleration, deceleration, and turning. By enabling continuous state and action spaces, DDPG provides a foundation for controlling vehicles in dynamic and high-stakes environments, making it well-suited for racing applications.

2.4 Autonomous Racing with Model-Based RL

While RL is effective in autonomous driving, alternative methods such as model predictive control (MPC) have also been explored for track navigation. Liniger et al. (2015) [4] presented an optimization-based approach to autonomous racing using MPC for scale RC cars. Although this approach is not based on RL, it highlights key challenges in optimizing a racing line, including precise control over acceleration and rotation. This work provides insights into how racing agents can balance competing objectives, a concept that informs the reward structure and safety constraints of RL-based racing agents.

2.5 Safe Reinforcement Learning for Autonomous Driving

Safety is a critical component in racing tasks, where maintaining control and staying within track boundaries is essential. Saxena et al. (2020) [6] explored safe reinforcement learning techniques for autonomous driving, focusing on avoiding obstacles and remaining on track. Although primarily targeted at general autonomous driving, the concept of safe RL is highly relevant to racing, where agents must learn to avoid actions that could lead to crashes or veering off track. This study serves as a reference for implementing safety constraints within the reward function to encourage responsible driving.

2.6 Reward Shaping in Simulated Racing Environments

In addition to these techniques, reward shaping has been used in racing applications to guide agents toward achieving optimal racing lines. Wurman et al.

(2002) [8] explored predictive state representations and reward shaping in racing, showing how carefully designed rewards can encourage the agent to balance speed with safe navigation. Reward shaping will play a key role in this project's RL framework, ensuring that the agent not only maximizes speed but also stays within track boundaries.

These works collectively underscore the complexity of using RL in racing environments, where agents must navigate high-dimensional, continuous control tasks under safety constraints. By drawing on these established techniques, this project seeks to develop an RL-based racing agent capable of navigating a simplified F1 track while balancing speed and safety, making it accessible as a foundational tool for aspiring racers.

3 Technical Approach

3.1 Environment Setup

To simulate the racing environment, we'll assume that the environment has an idealistic design where the track has a flat surface that can be represented using a 2-D matrix. This means that there are no elevation or bumps will be on the track and the width is assumed to be fixed across the track. We'll also ignore all types of friction like air friction and terrain. At each time step, the agent can select an action that affect 1 unit of acceleration in any directions. There is, however, a cap speed which limits how fast the agent can go. To further simplify the environment, all elements regarding the race car like tire degradation, inertia, momentum, etc. are also ignored.

3.2 Methodology

In this project, we'll incorporate a number of Reinforcement learning methods and algorithm and conduct experiments in our developed environment discussed above. We're aiming to use SARSA, Q-Learning, deep Q-Learning (DQL), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO) algorithms to train our agents.

3.2.1 Q-Learning

In Sutton and Barton[7], the off-policy Q-Learning update formula is given as follows:

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',a) - Q(S,A)]$$

However, the original Q-Learning requires storage of all the possible state in the Q-table leading to large consumption of memory resource.

3.2.2 Deep Q-Learning

To tackle the storage limitation of traditional Q-Learning, the Deep Learning element was introduced into learning and predicting the Q value. TD Target, which is the value used to train the neural network on, is defined as:

$$R + \gamma max_a Q(S', a)$$

To evaluate the training performance, TD error, formally defined as the difference between TD Target and the former Q-value estimation, is used as the loss function:

$$R + \gamma max_a Q(S', a) - Q(S, A)$$

3.2.3 Deep Deterministic Policy Gradient

DDPG is a combination of DQN and deterministic policy gradient. It has an actor network, which will handle predicting an action by learning the policy, and critic network, which will learn the value of Q and handle predicting next Q value[3].

Lilicrap et al, gives a formal updated of the actor policy by following sampled policy gradient [3]

$$\nabla_{\omega'} J \approx \frac{1}{N} \sum_i \nabla_\omega Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s|\theta^\mu)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}$$

And the critic network is updated by minimizing the loss function [3]

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^\mu))^2$$

Then it creates a copy of both network as target networks and update those using soft update $\theta \leftarrow \tau\theta + (1 - \tau)\theta'$ [3].

3.2.4 SARSA

For on-policy SARSA approach, the update formula is very much the same however, instead of measuring based on the optimal actions, it use the next action selected at the next state that it results in to calculate the differences. The formula was given by Sutton and Barton [7] as follows:

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', a') - Q(S, A)]$$

3.2.5 Proximal Policy Optimization

PPO is considered to be more efficient than SARSA due to its utilization of a network model to represent the policy. This allows it to be independent on the size of the Q-table. One characteristic of PPO is that it prevent large update on the policy due to its cliped objective function[1] which is defined below:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t\right)\right]$$

Where $r_t(\theta)$ is the ratio of change of the new value from the old.

3.3 Tasks

For our initial task, experiment will be carried out to determine the feasibility to train the agent in the given environment.

The racing car environment that comes with Gymnasium, a Python's library that comes with multiple predefined environment, will be utilized to escalate the development and testing time. There are two types of environment that is available in this package, discrete and continuous. In this initial work, we're mostly concern with discrete environment and will be using it to test the performance of our models.

Then, we'll train the agents using the techniques introduced in Sutton and Barton, Q-Learning and SARSA, as those are more straight forward to develop. We'll then evaluate the performance of the agent.

Once our initial work is fully functioning, we'll then looking into implementing more complex approaches DQL, DDPG, and PPO. We'll then evaluate the performance of these models using the same benchmark that we've used for Q-Learning and SARSA approaches and compare the result obtained.

If time permit, extensions will be made which are but not limited to adding environmental factors like tire degradation, elevation and obstacles, etc; usage of TAMER framework to introduce human reinforcement; or transforming the environment into that of a multi-agent.

4 Evaluation

The evaluation of this project will be centered on the agent's ability to optimize lap times and maintain track boundaries, with comparisons made across different reinforcement learning methods. The main evaluation criteria include lap time performance, track adherence, and generalization to multiple track layouts.

4.1 Lap Time Performance

Lap time will serve as a primary metric for evaluating each agent's efficiency in navigating the track. Rewards will be tied to lap time, either through full lap or sector times (to be determined), where a faster lap time results in higher rewards. By comparing cumulative rewards, we can measure the overall effectiveness of each method in minimizing lap time. This approach allows us to quantify each agent's speed optimization and observe improvements over training episodes.

4.2 Track Boundary Adherence

Each agent will incur penalties for going out of bounds, emphasizing the importance of maintaining control while optimizing speed. The boundary adherence metric will help determine the agent's consistency in staying within the track limits, which is crucial for real-world applicability. Frequent boundary violations or excessively aggressive maneuvers will be penalized, guiding the agent toward safer driving behaviors.

4.3 Method Comparison

To evaluate the effectiveness of various reinforcement learning algorithms, each agent will be trained and tested using different methods, such as DQN, DDPG, and potentially others like Q-learning. The reward progression over time for each method will be tracked and compared, providing insights into the learning stability, convergence rates, and overall efficiency of each approach.

4.4 Track Generalization

To assess each agent's adaptability, evaluations will be conducted on three different race tracks with relatively simple layouts: Monza, Red Bull Ring, and Fuji Speedway. These tracks were chosen for their distinct characteristics, allowing us to test the agent's ability to generalize its learned racing line strategies across varied track configurations. Performance on each track will be measured independently, focusing on lap times and boundary adherence. This multi-track evaluation will help identify the robustness of each method in handling diverse racing environments.

By analyzing these metrics across different algorithms and track layouts, this evaluation will provide a comprehensive view of each method's effectiveness in race line optimization, stability, and adaptability.

5 Individual Responsibilities

5.1 Joseph Hadidjojo

1. Environmental Setup
2. Implementing and evaluate SARSA
3. Implementing and evaluate PPO
4. Final Report

5.2 Tung Pham

1. Environmental Setup
2. Implementing and evaluate Q-Learning
3. Implementing and evaluate DQN
4. Implementing and evaluate DDPG
5. Final Report

References

[1] Hugging Face. The deep q-learning algorithm. `https://huggingface.co/learn/deep-rl-course/unit3/deep-q-learning`, 2023. Hugging Face Deep RL Course.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[4] Alexander Liniger, John Lygeros, and Pierre Apkarian. Optimization-based autonomous racing of 1: 43 scale rc cars. In *2015 European Control Conference (ECC)*, pages 586–591. IEEE, 2015.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[6] Abhinav Saxena, Phillip Isola, et al. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:2004.06477*, 2020.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018.

[8] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Racing against opponents with approximate predictive state representations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 523–528. Citeseer, 2002.