Reinforcement Learning for Collision Avoidance in Autonomous Driving

Nazmus Sakib
Department of Computer Science
University of Colorado Colorado Springs
nsakib@uccs.edu

Abstract

Reinforcement Learning (RL) for dynamic collision avoidance in autonomous driving has gained popularity in recent years. This research intends to evaluate various RL algorithms for dynamic collision avoidance in autonomous driving. After reviewing several papers, we discovered five common reinforcement learning methods: Q-Learning, Deep Q-Learning, SARSA, MLP SARSA, and PPO. Therefore, the main goal of this research is to assess how well these five reinforcement learning algorithms in autonomous driving across a busy roadway. Additionally, we plan to train these five models in a dynamic collision avoidance urban simulation environment to evaluate their performances.

Introduction

A fundamental concept of ensuring the safety of drivers and their vehicles is collision avoidance (Yurtsever et al., 2020). Due to rising vehicle density, autonomous vehicle navigation is becoming a difficult issue. There are two types of obstacles that can be found on roads: static obstacles, such as cars parked there, and dynamic obstacles, which include moving objects like animals that move randomly (Almazrouei et al., 2023). Autonomous driving requires the ability to detect and avoid both static and dynamic obstacles. To ensure safety, various sensors, including vision, ultrasonic radar, and lidar, need to be used. Vision sensors can distinguish and classify obstacles, but they cannot precisely determine their distance for action. Combining ultrasonic radar or lidar data with a vision sensor improves obstacle detection accuracy in various settings. Supervised learning outperforms traditional approaches in detecting objects using vision and ultrasonic radar. However, this strategy requires extensive ground truth data from various situations to train the machine learning model (Arvind and Senthilnath, 2019). To overcome this limitation, combining supervised and reinforcement learning may eliminate the need to manually provide ground truth for obstacles (Babu et al., 2016). Reinforcement learning (RL) is a machine learning technique that allows an agent, such as an autonomous car, to learn its surroundings based on prior actions, states, and rewards (Kaelbling et al., 1996). In recent years, there has been an increase in the use of RL for dynamic collision avoidance in autonomous driving (Kim et al., 2020).

The goal of this research is to assess several reinforcement learning algorithms for dynamic collision avoidance in autonomous driving. We identified five widely used reinforcement learning techniques: Q-Learning, Deep Q-Learning, SARSA, MLP SARSA, and PPO. Thus, the primary objective of this study is to evaluate the effectiveness of these five reinforcement learning techniques for autonomous driving on a busy road.

Background

Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning technique in which an agent functions in a given environment and tries to find a policy (π) that maximizes a cumulative reward function. The policy specifies what action should be done, a, in a state, s. The environment will thereafter shift to a new state, s', and yield a reward, r.

Q-Learning

The agent in Q-Learning, a reinforcement learning algorithm, tries to learn $Q^*(s, a)$, the optimal action-value function. The maximum expected return that results from being in a state (s), acting (a), and then according to the best course of action (π) is the definition of this function.

Deep Q-Learning

Deep Q-Learning is an advanced reinforcement learning technique that integrates deep neural networks with Q-learning, a value-based method for determining the optimal action-selection policy. In Deep Q-Learning, a deep neural network, known as the Q-network, is used to approximate the Q-value function, which estimates the value of taking a given action in a particular state. This approach allows the agent to handle complex, high-dimensional environments that traditional Q-learning methods struggle with due to their tabular nature. For a state-action pair (s, a), the Q-value is updated as:

$$\mathrm{Q}_{new}(s,a) \leftarrow (1-\alpha)\cdot Q(s,a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$$

State Action Reward State Action (SARSA)

The State-Action-Reward-State-Action (SARSA) algorithm is a fundamental approach in reinforcement learning that operates on the principle of learning an action-value function to guide the actions of an agent within an environment. Unlike Q-Learning, which is based on the principle of maximizing future rewards, SARSA takes a more conservative approach by considering the actual next action the agent will take according to its current policy. In essence, SARSA updates its action-value function, $Q(s, a)$, based on the observed transition from the current state-action pair (s, a) to the next state-action pair (s', a') and the reward received in the process.

Multi-Layer Perceptron (MLP) based State-Action-Reward-State-Action (SARSA)

The Multi-Layer Perceptron (MLP) based State-Action-Reward-State-Action (SARSA) reinforcement learning method integrates the classic SARSA algorithm with the power of neural networks, specifically MLPs, to handle environments with high-dimensional state spaces. In this approach, the MLP is used to approximate the action-value function $Q(s, a)$, enabling the agent to learn and generalize across a vast number of states and actions. By inputting the state (and possibly the action) into the MLP, the network outputs the estimated values of taking each action in the given state, effectively learning the optimal policy through iterative updates. The SARSA update rule is applied in a way that the target for the MLP's output is adjusted based on the reward received and the estimated value of the next state-action pair, $Q(s', a')$, according to the policy being followed.

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) aims to improve learning stability and efficiency in reinforcement learning by controlling the size of policy updates. It does so through an objective function that encourages the new policy to stay close to the old policy, ensuring that the updates are not too large, which could destabilize training (Wei et al., 2019). For an action a taken in state s, the policy ratio is given by:

$$\mathrm{r}(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$$

Related Work

We examine several studies on the application of reinforcement learning methods to autonomous driving. A couple of related works are listed below.

(Babu et al., 2016) have developed an autonomous agent that leverages Q-learning methods to find the shortest path from a current state to a desired state in a fully dynamic environment based on camera data. Similarly, (Hong et al., 2017) have used a fuzzy Q-learning method to identify obstacles in a dynamic environment utilizing input from ultrasonic sensors. The State Action Reward State Action (SARSA) method was employed by (Rais et al., 2023) to avoid collisions in autonomous vehicles on highways. In an urban dynamic environment scenario, (Arvind and Senthilnath, 2019) have employed the State Action Reward State Action (SARSA) method using Multi Layer Perception (MLP) for autonomous vehicle obstacle detection and avoidance. (Arvind and Senthilnath, 2020) employed a policy-free, model-free Q-learning based reinforcement learning algorithm with a multi-layer perceptron neural network (MLP-NN) to forecast the best course of action for the vehicle in the future, given its present state.

In contrast to earlier studies, the objective of this research is to evaluate the performance of reinforcement learning algorithms for driving autonomously over a busy road. In addition, we intend to assess the performance of these popular reinforcement learning methods by training them in a dynamic collision avoidance urban simulation scenario.

Proposed Methodology and Environment Setup

The objective of our research is to develop and refine the capabilities of an autonomous car agent, enabling it to navigate through complex urban environments, specifically across densely populated highways, while avoiding any collisions with other vehicles. This shows the autonomous agent's ability to dynamically adjust its speed and positioning on the driving, leveraging a series of discrete actions tailored to navigate towards its designated destination location with optimal efficiency. To achieve this, we have implemented the principles of both Q-Learning and SARSA—two pivotal, model-free reinforcement learning algorithms. These algorithms are instrumental in guiding the agent towards developing an optimal action-selection policy. Through a process characterized by both exploration of the environment and exploitation of acquired knowledge, the agent incrementally maximizes its cumulative reward. This is achieved by continuously updating its understanding and valuation of state-action pairings (Q-values) across a multitude of simulation episodes.

Environment Setup:

For the purpose of our experiments, we have aimed to create a simulation that mirrors the complexities of an urban traffic system, filled with dynamic obstacles. We plan to utilize the pygame package (Gym and Sanghi, 2021) to craft a detailed simulation environment that can accurately represent these challenges. However, for the scope of this midterm report and the current stage of our experimental setup, we have opted for a simplified model to conduct our simulations. This model takes the form of a 4x10 grid, conceptualized to emulate a highway scenario where each column represents a lane on the highway and each row symbolizes different positions within those lanes. For our simulation, we have assumed that all traffic flows unidirectionally, and we have placed various other cars (acting as static obstacles) throughout this virtual highway to simulate real-life driving conditions.

States: The state space is defined by a three-dimensional grid representing the environment in which the agent car operates. Each state is a tuple (row, column, speed), where: row and column define the agent's position in the grid, with the grid having 10 rows and 4 columns and speed represents the agent's speed level, discretized into three levels: low

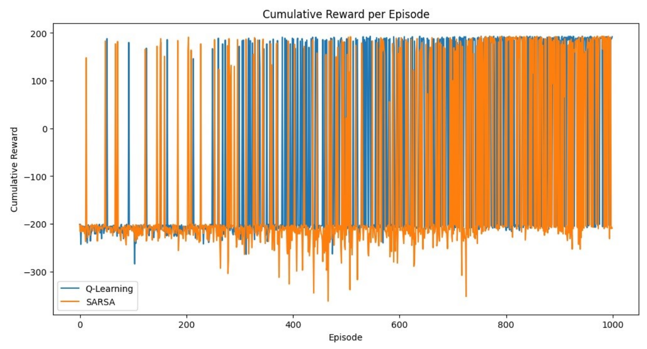


Figure 1: Comparisons result between Q-Learning and SARSA based on Cumulative Reward Over Episodes for navigating from start to goal location.

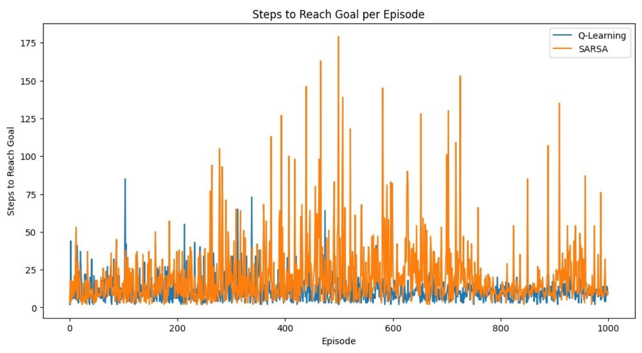


Figure 2: Comparisons result between Q-Learning and SARSA based on Number of Steps to Reach Goal for navigating from start to goal location.

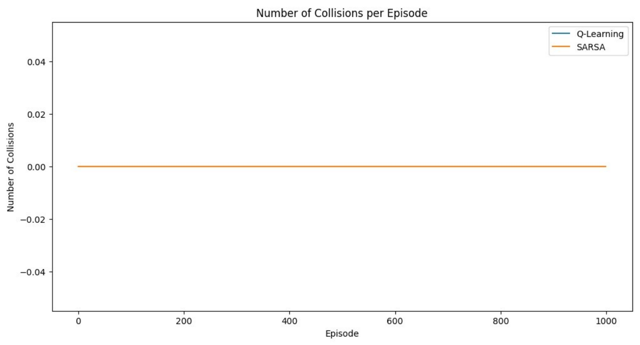


Figure 3: Comparisons result between Q-Learning and SARSA based on Number of Collisions for navigating from start to goal location.

(0), medium (1), and high (2). The total state size is the product of the number of rows, columns, and speed levels, resulting in a 120 distinct states (10 rows * 4 columns * 3 speed levels).

Actions: The agent can performs one of five actions at any given state:

- **Forward:** Move forward in the grid, with the distance moved dependent on the current speed level.
- **Lane Change Left:** Move one column to the left, if not already in the leftmost column.
- **Lane Change Right:** Move one column to the right, if not already in the rightmost column.
- **Accelerate:** Increase the speed level by one, unless already at the maximum speed.
- **Decelerate:** Decrease the speed level by one, unless already at the minimum speed.

Rewards: The reward structure is designed to guide the agent car towards the goal while avoiding obstacles:

- **Moving Forward:** A default reward of -1.0 is assigned to encourage the agent to reach the goal with as few steps as possible.
- **Collision:** A significant penalty of -200.0 is assigned if the agent collides with an obstacle (other cars) at any speed, discouraging collisions.
- **Reaching the Goal:** A reward of 200.0 is given for reaching the goal state, incentivizing the agent to navigate towards the goal.

Parameter Selection, Running Experiments, Simulation Results, and Discussion

Parameter Selection:

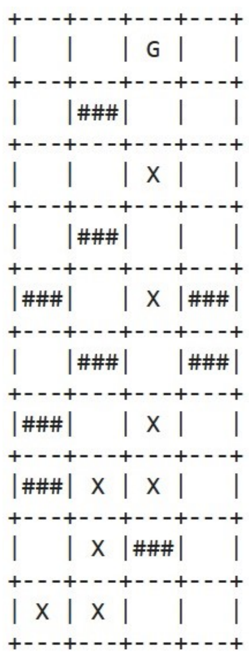
Parameter selection is a crucial step in setting up reinforcement learning algorithms because the chosen parameters significantly influence the learning rate, the balance between exploration and exploitation, and the convergence towards an optimal policy. For our autonomous car navigating a grid-like representation of a highway, the following parameters are considered:

- **Learning Rate:** We consider 0.1 as a starting learning rate, allowing the agent to update its Q-values by considering 10% of new learning while retaining 90% of the old value.
- **Discount Factor:** We consider 0.95 as a discount factor.
- **Exploration Rate and Exploration Decay Rate:** Initially we consider 1.0 as a exploration rate but later we introduce a decay rate of 0.001 which ensures a gradual shift from exploration to exploitation.

Running Experiments:

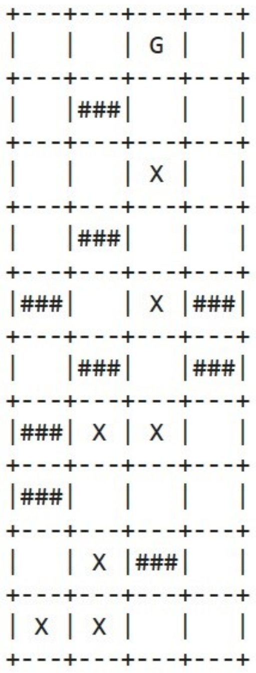
Running experiments involves using the selected parameters to train the agent over a series of episodes, allowing it to learn from its environment and adjust its policy accordingly. Here are how we typically unfolds:

- **Number of Episodes:** The total number of episodes for training significantly impacts the agent's performance. Each episode provides an opportunity for the agent to explore the environment, make decisions, and learn from the outcomes. For the given setup, we consider 50,000



```
+---+---+---+---+
|   |   | G |   |
+---+---+---+---+
|   |###|   |   |
+---+---+---+---+
|   |   | X |   |
+---+---+---+---+
|   |###|   |   |
+---+---+---+---+
|###|   | X |###|
+---+---+---+---+
|   |###|   |###|
+---+---+---+---+
|###|   | X |   |
+---+---+---+---+
|###| X | X |   |
+---+---+---+---+
|   | X |###|   |
+---+---+---+---+
| X | X |   |   |
+---+---+---+---+
```

Figure 4: Path grid of SARSA agent after reaching convergence.



```
+---+---+---+---+
|   |   | G |   |
+---+---+---+---+
|   |###|   |   |
+---+---+---+---+
|   |   | X |   |
+---+---+---+---+
|   |###|   |   |
+---+---+---+---+
|###|   | X |###|
+---+---+---+---+
|   |###|   |###|
+---+---+---+---+
|###| X | X |   |
+---+---+---+---+
|###|   |   |   |
+---+---+---+---+
|   | X |###|   |
+---+---+---+---+
| X | X |   |   |
+---+---+---+---+
```

Figure 5: Path grid of Q-Learning agent after reaching convergence.

episodes which allow the agent sufficient interactions to learn an effective policy for navigating from the start to the goal while avoiding obstacles.

- **Per-Episode Process:** Within each episode, the agent starts from the predefined starting point and makes decisions at each step based on either exploration (random choice) or exploitation (choosing the best-known action). The outcomes of these actions (new state and reward) are then used to update the Q-values.
- **Q-Value Update:** After each action, the Q-value for the state-action pair is updated, incorporating the immediate reward and the discounted value of the next best/randomly select any action. This process iteratively improves the agent's strategy.

Simulation Results and Discussion:

To evaluate the performance of the Q-Learning and SARSA algorithms, we consider following three performance evaluation matrices:

Cumulative Reward per Episode: This metric sums up all the rewards obtained by the agent in each episode. By plotting the cumulative reward per episode for both Q-Learning and SARSA, we assess how effectively each algorithm maximizes rewards over time.

Number of Steps to Reach Goal: This metric counts the number of steps the agent takes to reach the goal in each episode. Plotting the number of steps per episode for both algorithms highlights their efficiency improvements or declines over time.

Number of Collisions: This metric records the number of times the agent collides with other cars (static obstacles) in each episode. By observing the number of collisions per episode, we evaluate the safety of the navigation strategies learned by the agent using these two algorithms.

To visually compare the performance efficiency between Q-Learning and SARSA algorithms, we create three Plots that show the Cumulative Reward Over Episodes, Number of Steps to Reach Goal, and Number of Collisions for both algorithms. Figure 1 illustrates the performance comparisons between Q-Learning and SARSA based on Cumulative Reward Over Episodes where Figure 2 illustrates the performance comparisons between Q-Learning and SARSA based on Number of Steps to Reach Goal and Figure 3 illustrates the performance comparisons between Q-Learning and SARSA based on Number of Collisions.

From the results, it can easily be seen that, in terms of Cumulative Reward, Q-Learning demonstrates a more aggressive approach towards maximizing rewards, possibly outperforming SARSA in achieving higher cumulative rewards over episodes. Additionally, when evaluating the Steps to Reach Goal, Q-Learning potentially finds shorter or more efficient paths to the goal compared to SARSA. SARSA, being more conservative due to its on-policy nature, takes into account the current policy's actions, which may result in slightly longer paths as it avoids potential penalties more cautiously. However, regarding the Number of Collisions, SARSA exhibits fewer collisions over time compared to Q-Learning. SARSA's cautious strategy, which considers the consequences of actions based on the current policy, tends to avoid risky moves that could lead to collisions. Figure 4 and 5 illustrate the grid path of the SARSA and Q-Learning agent after reaching convergence.

Limitations of Our Experiments :

Thus far, we have only implemented two RL algorithms (Q-Learning and SARSA) in our experiments. Other algorithms (Deep Q-Learning, MLP-based SARSA, and PPO) need to be implemented. Our next goal is to implement other algorithms along with these two algorithms and present the results in our final submission. In addition, we aim to incorporate animation and dynamic obstacles into our implementation to make the experiment mimic a real-world scenario.

Conclusion

To achieve dynamic collision avoidance in autonomous vehicles, this study aims to examine five reinforcement learning (RL) algorithms: Q-Learning, Double Q-Learning, SARSA, MLP-based SARSA, and PPO. The main goal is to assess critically how well these algorithms work for safely driving autonomous cars across busy highways. Through the use of these RL methods in a simulated urban collision avoidance scenario, the research aims to provide significant insights into how well they perform on one another.

Timeline

This is a general timeline that shows how we think the project will advance.

Table 1: Project Milestones and Timelines

Date	Milestone
4/1/2024	Environment Setup for complex urban simulation.
4/15/2024	Train RL agents using complex urban simulation environment.
4/25/2024	Update and improve RL agents, compare with others, comparisons table generation.
5/1/2024	Complete Final version of project and report.

References

[Almazrouei et al., 2023] Almazrouei, K., Kamel, I., and Rabie, T. (2023). Dynamic obstacle avoidance and path planning through reinforcement learning. *Applied Sciences*, 13(14):8174.

[Arvind and Senthilnath, 2019] Arvind, C. and Senthilnath, J. (2019). Autonomous rl: Autonomous vehicle obstacle avoidance in a dynamic environment using mlp-sarsa reinforcement learning. In *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, pages 120–124. IEEE.

[Arvind and Senthilnath, 2020] Arvind, C. and Senthilnath, J. (2020). Autonomous vehicle for obstacle detection and avoidance using reinforcement learning. In *Soft Computing for Problem Solving: SocProS 2018, Volume 1*, pages 55–66. Springer.

[Babu et al., 2016] Babu, V. M., Krishna, U. V., and Shahensha, S. (2016). An autonomous path finding robot using q-learning. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–6. IEEE.

[Gym and Sanghi, 2021] Gym, O. and Sanghi, N. (2021). *Deep reinforcement learning with python*. Springer.

[Hong et al., 2017] Hong, J., Tang, K., and Chen, C. (2017). Obstacle avoidance of hexapod robots using fuzzy q-learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[Kim et al., 2020] Kim, M., Lee, S., Lim, J., Choi, J., and Kang, S. G. (2020). Unexpected collision avoidance driving strategy using deep reinforcement learning. *IEEE Access*, 8:17243–17252.

[Rais et al., 2023] Rais, M. S., Boudour, R., Zouaidia, K., and Bougueroua, L. (2023). Decision making for autonomous vehicles in highway scenarios using harmonic sk deep sarsa. *Applied Intelligence*, 53(3):2488–2505.

[Wei et al., 2019] Wei, H., Liu, X., Mashayekhy, L., and Decker, K. (2019). Mixed-autonomy traffic control with proximal policy optimization. In *2019 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE.

[Yurtsever et al., 2020] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K. (2020). A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469.