Automation and Machine Learning (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/autml.2026.070101
ISSN 2516-5003 Vol. 7 Num. 1

A Comparative Study of Basic Reinforcement Learning Algorithms for Two-Wheeled Mobile Robot Path Tracking

Hongyuan Liu

School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China
3102772267@qq.com

Keywords: Reinforcement Learning; Two-Wheeled Mobile Robot; Path Tracking; Gym/Gazebo; Algorithm Comparison; Convergence Speed; Robustness

Abstract: To address the adaptability problem of two-wheeled mobile robots in path tracking tasks under different scenarios and improve the accuracy and robustness of robot motion control, this paper builds a path tracking simulation platform for two-wheeled mobile robots based on Gym and Gazebo. Three classic basic reinforcement learning algorithms, namely Q-Learning, Sarsa, and Deep Q-Network (DQN), are selected for comparative experimental research. Three map scenarios with varying complexity—simple, complex, and dynamic obstacle environments—are designed to conduct quantitative analysis and qualitative evaluation of the algorithms' performance from three core dimensions: convergence speed, control stability, and environmental robustness. Experimental results show that the Q-Learning algorithm converges fastest in simple static scenarios but has insufficient robustness; the Sarsa algorithm exhibits superior safe exploration capabilities; and the DQN algorithm demonstrates remarkable adaptive advantages in complex dynamic scenarios. This paper finally provides clear algorithm selection recommendations for different application scenarios, offering theoretical reference and technical support for the engineering practice of two-wheeled mobile robot path tracking control.

1. Introduction

With the wide application of mobile robot technology in fields such as logistics and distribution, service robots, and industrial inspection, the two-wheeled mobile robot, as the core carrier of mobile robots, has seen its path tracking accuracy and adaptability become key factors restricting system performance. Traditional path tracking control methods, such as PID control and model predictive control, rely on accurate robot kinematic models and prior environmental information. They are prone to problems like decreased control accuracy and insufficient robustness in complex, variable, or unknown environments. As a model-free machine learning method, reinforcement learning enables agents to obtain optimal control strategies through interactive trial-and-error with the environment without relying on precise mathematical models[5]. It has shown unique advantages in autonomous decision-making tasks in unknown environments, providing a new solution for two-wheeled mobile robot path tracking control[6].

At present, basic reinforcement learning algorithms have made certain progress in mobile robot control applications, but systematic comparative research on algorithm performance under scenarios with varying complexity remains to be improved. As classic tabular reinforcement learning algorithms, Q-Learning and Sarsa feature simple principles and convenient implementation, and are widely used in control tasks with low-dimensional state spaces . Deep Q-Network (DQN) fits the Q-value function through deep neural networks, breaking through the curse of dimensionality faced by tabular algorithms in high-dimensional state spaces, and is suitable for more complex control scenarios[1,2]. Existing studies have mainly focused on the application of individual algorithms, while systematic comparative research and conclusion summaries on the performance of different algorithms under scenarios with varying complexity are still lacking.

Based on this, this paper builds a two-wheeled mobile robot path tracking simulation platform based on Gym and Gazebo, designs map scenarios with different complexity levels, and conducts a comprehensive comparative study of Q-Learning, Sarsa, and DQN algorithms from three dimensions: convergence speed, stability, and robustness. This study clarifies the advantages and limitations of each algorithm and provides algorithm selection recommendations for specific scenarios, offering theoretical basis and practical guidance for algorithm selection and engineering application in two-wheeled mobile robot path tracking control.

2. Related Theories and Simulation Platform Construction

2.1. Basic Reinforcement Learning Theories

The core framework of reinforcement learning consists of five elements: agent, environment, state, action, and reward. The agent executes corresponding actions by perceiving the environmental state; the environment feeds back the new state and reward signal according to the agent's actions; the agent continuously optimizes the action strategy to maximize the cumulative reward, and finally forms the optimal control strategy.

Q-Learning is an off-policy tabular reinforcement learning algorithm. Its core mechanism is to iteratively update the action-value function (Q-value) based on the Bellman equation. A key feature of this algorithm is that it does not need to follow the current exploration strategy when updating the Q-value, and can optimize the target strategy by using historical interaction data stored in experience replay. The update process adjusts the current Q-value based on three key parameters: learning rate, which controls the step size of each update; immediate reward, which reflects the effectiveness of the current action; and discount factor, which balances the importance of immediate reward and future cumulative reward .

Sarsa is an on-policy tabular reinforcement learning algorithm, and its core difference from Q-Learning lies in the consistency between the Q-value update process and the exploration strategy. When updating the Q-value, Sarsa needs to wait for the agent to actually select and execute the next action under the current strategy, and then completes the update based on the next action and the corresponding reward signal. This on-policy characteristic makes Sarsa pay more attention to the safety of the exploration process, effectively avoiding dangerous exploration behaviors that may occur in off-policy algorithms, thus being more suitable for control tasks with high safety requirements .

DQN algorithm integrates deep neural networks into the Q-Learning framework, which solves the problem that tabular algorithms are difficult to apply in high-dimensional state spaces. It uses deep neural networks to fit the Q-value function, converting the discrete Q-value table lookup into continuous function approximation. At the same time, DQN introduces two key mechanisms to improve training stability: one is the experience replay mechanism, which stores the interaction data between the agent and the environment in a buffer, and randomly samples data batches for training to reduce the correlation between data; the other is the dual-network mechanism, which separates the current network for Q-value prediction and the target network for target Q-value calculation, and periodically updates the parameters of the target network to avoid training oscillation.

2.2. Kinematic Model of Two-Wheeled Mobile Robot

The two-wheeled mobile robot studied in this paper adopts a differential drive mode, and its kinematic model is constructed based on the assumption of planar rigid body motion, ignoring non-ideal factors such as wheel slip and ground friction in actual operation. The pose of the robot is determined by three key parameters: horizontal coordinate, vertical coordinate, and heading angle. The movement of the robot is driven by the speed difference between the left and right wheels, where the linear velocity of the robot's center of mass and the angular velocity of the heading rotation are jointly determined by the rotational speeds of the left and right wheels, the wheel radius, and the wheelbase (the distance between the two wheels). Specifically, the linear velocity of the robot is positively related to the sum of the rotational speeds of the left and right wheels, while the angular velocity of the heading is positively related to the difference between the rotational speeds of the left and right wheels.

In the path tracking task, the state space of the robot mainly includes two types of information: the current pose parameters and the deviation from the target path. The deviation information is specifically divided into position deviation (the linear distance between the robot's center of mass and the target path) and heading deviation (the angle difference between the robot's current heading and the tangent direction of the target path). The action space is defined as the discrete control quantity of the rotational speeds of the left and right wheels. By adjusting the combination of the rotational speeds of the two wheels, the robot can realize basic movements such as straight-line driving, left turning, and right turning, thereby adjusting its pose in real time to complete the path tracking task.

2.3. Simulation Platform Construction

This paper builds a two-wheeled mobile robot path tracking simulation platform through the combination of Gym and Gazebo, and the two tools play complementary roles[3]. Gym provides a standardized reinforcement learning task interface, which is mainly used to define the boundary conditions of the path tracking task, including the specific scope of the state space and action space, and the design of the reward function. Gazebo provides a high-precision three-dimensional physical simulation environment, which is responsible for constructing the robot's physical model, building different map scenarios, and simulating real physical rules such as gravity, friction, and collision.

The key hardware parameters of the simulation platform are set as follows: the wheel radius of the robot is 0.05 meters, the wheelbase is 0.15 meters, and the maximum rotational speed of a single wheel is 2 radians per second. To test the adaptability of the algorithm under different environmental complexities, three types of map scenarios are designed: the simple map is a rectangular open area without any obstacles, and the target path is a simple combined trajectory composed of straight lines and small-radius arcs; the complex map adds multiple static obstacles (such as cylinders and cubes) on the basis of the simple map, and the target path is designed as a complex curve with multiple inflection points, requiring the robot to frequently adjust its heading; the dynamic obstacle map further adds randomly moving obstacles (with variable moving speed and direction) on the basis of the complex map to simulate the dynamically changing actual application environment.

The design of the reward function is crucial to the convergence effect of the reinforcement learning algorithm. This paper adopts a mixed reward mechanism combining dense reward and sparse reward to balance the training efficiency and control accuracy. Specifically, during the movement of the robot, if it can move along the target path and the position deviation and heading deviation are within the preset allowable range, a continuous positive reward is given; if the deviation exceeds the threshold or a collision with an obstacle occurs, a large negative reward is given to punish the unsafe behavior; when the robot successfully completes the entire path tracking task (reaches the end point of the target path), a one-time large terminal reward is given. By reasonably setting the weight ratio between different types of rewards, the agent is guided to learn the optimal path tracking strategy efficiently while avoiding dangerous behaviors.

3. Experimental Design and Result Analysis

3.1. Experimental Parameter Setting

To ensure the fairness and effectiveness of the comparative experiment, the core hyperparameters of the three algorithms are uniformly optimized and set. The specific parameter values are as follows: the learning rate, which controls the update step size of the algorithm, is set to 0.1; the discount factor, which balances immediate and future rewards, is set to 0.95; the exploration rate, which controls the probability of the agent selecting random actions, adopts an exponential decay strategy—starting from an initial value of 1.0 (full exploration in the early stage), decaying at a rate of 0.995 per iteration, and finally stabilizing at 0.01 (maintaining a small probability of exploration in the later stage to avoid local optimality). For the DQN algorithm, the neural network structure adopts a three-layer fully connected network: the input layer dimension is consistent with the state space dimension, the number of nodes in the two hidden layers is 128 and 64 respectively, and the output layer dimension is consistent with the action space dimension. The optimizer selects Adam, the learning rate is set to 0.001, the capacity of the experience replay buffer is 10,000, and the batch size for each network training is 32.

Three core evaluation indicators are established to comprehensively compare the performance of the three algorithms: (1) Convergence speed: measured by the number of training iterations required for the algorithm to reach a stable tracking accuracy (i.e., the path deviation remains within a small range and no longer fluctuates significantly). A smaller number of iterations indicates a faster convergence speed. (2) Control stability: evaluated by the root mean square error (RMSE) of the path deviation during the stable tracking phase. A smaller RMSE value indicates that the robot's movement trajectory is closer to the target path and the control is more stable. (3) Environmental robustness: measured by two indicators, the task completion rate (the proportion of successful path tracking without collision in multiple test rounds) and the tracking accuracy retention ability (the degree of increase in RMSE compared with the stable phase in the simple scenario) when the robot is in a dynamic or complex environment. A higher task completion rate and a smaller increase in RMSE indicate stronger robustness.

3.2. Experimental Results in Simple Map Scenarios

In the simple map scenario, all three algorithms can successfully complete the path tracking task, but there are obvious differences in convergence speed and control stability. Among them, the Q-Learning algorithm has the fastest convergence speed, requiring only about 800 training iterations to reach stable tracking accuracy, and the RMSE of the path deviation in the stable phase is 0.025 meters. The convergence speed of the Sarsa algorithm is slightly slower, requiring about 1,200 iterations to stabilize, but its control stability is better, with a path deviation RMSE of 0.022 meters. The DQN algorithm has the slowest convergence speed, requiring about 3,000 iterations to reach a stable state. This is mainly due to the need for repeated training and parameter adjustment of the deep neural network. However, due to the high fitting accuracy of the neural network, the path deviation RMSE of the DQN algorithm in the stable phase is the smallest, at 0.020 meters.

The root cause of the above performance differences lies in the characteristics of the algorithm itself and the matching degree with the scenario. In the simple map scenario, the state space dimension is low, and the tabular algorithm can quickly traverse all possible state-action combinations and update the Q-value to obtain the optimal strategy through iterative learning. The off-policy characteristic of Q-Learning allows it to use all historical exploration data (including data generated by non-optimal strategies) to update the target strategy, thus having higher learning efficiency and faster convergence speed than Sarsa. The on-policy characteristic of Sarsa requires that the update of the Q-value must be based on the actions actually selected by the current strategy, which makes it more cautious in the exploration process, avoiding unnecessary path deviations caused by radical exploration, so its control stability is slightly better than that of Q-Learning. For the DQN algorithm, the low-dimensional state space cannot fully exert the advantage of the neural network in processing high-dimensional information. On the contrary, the complexity of network training (such as the risk of overfitting and training oscillation) leads to a slow convergence speed. However, the continuous function approximation ability of the neural network makes its fitting accuracy of the Q-value higher, so the final tracking error is the smallest.

3.3. Experimental Results in Complex Map Scenarios

In the complex map scenario with multiple static obstacles, the robot needs to frequently adjust its heading angle to avoid obstacles while ensuring that it does not deviate from the target path. The performance differences between the three algorithms are further amplified. The convergence speed of the Q-Learning algorithm decreases significantly, requiring about 2,500 training iterations to stabilize. More importantly, during the tracking process, the Q-Learning algorithm is prone to excessive path deviation near obstacles, and the RMSE of the path deviation in the stable phase increases to 0.068 meters. In some training rounds, collisions with obstacles even occur, resulting in task failure. The Sarsa algorithm performs better in this scenario: its convergence speed is 1,800 iterations, the path deviation RMSE in the stable phase is 0.045 meters, and the task completion rate reaches 95%. This is due to the safe exploration characteristic of Sarsa, which can effectively avoid obstacles by predicting the potential risks of the next action. The DQN algorithm still has a slow convergence speed (about 3,500 iterations), but it shows excellent environmental adaptability. Its path deviation RMSE in the stable phase is only 0.030 meters, and the task completion rate reaches 100%, which is significantly better than the two tabular algorithms.

The main reason for this performance gap is the impact of the curse of dimensionality on tabular algorithms. In complex map scenarios, the introduction of obstacles increases the dimension of the state space (the robot needs to simultaneously perceive the position of obstacles and the deviation from the target path), making the Q-value table of tabular algorithms expand exponentially. This not only increases the storage pressure but also significantly reduces the efficiency of Q-value update, leading to a decline in algorithm performance. The aggressive exploration strategy of Q-Learning makes it more likely to fall into local optimality or collide with obstacles when facing multiple obstacles. Although the Sarsa algorithm can reduce collision risks through conservative exploration, its tabular structure still cannot effectively process high-dimensional state information, resulting in a certain degree of attenuation in tracking accuracy. The DQN algorithm solves the problem of the curse of dimensionality through deep neural networks. The neural network can automatically extract the key features of the complex environment (such as the relative position of obstacles and the trend of the target path) from the high-dimensional state space, and fit the Q-value function more accurately, thus maintaining excellent tracking performance in complex scenarios.

3.4. Experimental Results in Dynamic Obstacle Map Scenarios

In the dynamic obstacle map scenario where obstacles move randomly, the environmental state is highly uncertain, and the robustness differences between the three algorithms are most significant. The Q-Learning algorithm has the worst performance: its task completion rate is only 60%, and the path deviation RMSE in the test phase increases to 0.12 meters, which is nearly five times that in the simple scenario, indicating that it cannot adapt to the dynamic changes of the environment. The performance of the Sarsa algorithm is slightly better, with a task completion rate of 75% and a path deviation RMSE of 0.08 meters, but its tracking accuracy still attenuates significantly compared with the simple scenario. The DQN algorithm shows obvious advantages in robustness: its task completion rate remains above 90%, and the path deviation RMSE is 0.045 meters, which is only a small increase compared with the stable phase in the simple scenario, indicating that it can effectively adapt to the dynamic changes of the environment.

The fundamental reason for this difference is the different ways of processing environmental changes between tabular algorithms and DQN algorithms. Tabular algorithms rely on fixed state-action Q-value tables to make decisions. When the environment changes dynamically (such as the movement of obstacles leading to changes in the state corresponding to the same position), the original Q-value table can no longer accurately reflect the current environmental state, resulting in the failure of the control strategy. The DQN algorithm adopts an end-to-end learning method. Through the continuous interaction with the dynamic environment, the neural network can continuously update the parameters to learn the dynamic characteristics of the environment (such as the movement law of obstacles). At the same time, the experience replay mechanism of DQN can store interaction data under different environmental states (including static and dynamic states), which enhances the generalization ability of the algorithm to environmental changes. When facing new dynamic obstacle positions, the DQN algorithm can quickly adjust the control strategy based on the learned general rules, thus maintaining stable tracking performance.

4. Algorithm Selection Recommendations and Discussion

4.1. Algorithm Selection Recommendations for Specific Scenarios

Based on the comprehensive experimental results and the inherent characteristics of the three algorithms, this paper puts forward targeted algorithm selection recommendations for different application scenarios:

(1) Simple static scenarios: The Q-Learning algorithm should be preferred. In such scenarios, the environmental structure is simple, there are no obstacles or only a small number of fixed obstacles, and the state space dimension is low. The Q-Learning algorithm has the advantages of fast convergence speed, simple implementation logic, and low computational resource consumption, which can ensure high-efficiency and high-precision path tracking. Typical application scenarios include indoor fixed-path logistics distribution, simple factory floor inspection, and other tasks that require fast response and stable environmental conditions.

(2) Safety-priority static complex scenarios: The Sarsa algorithm should be preferred. In such scenarios, there are multiple static obstacles, and the movement space of the robot is limited. The risk of collision is high, and safety is more important than convergence speed. The on-policy safe exploration characteristic of the Sarsa algorithm can effectively predict the potential risks of the next action, avoid radical exploration behaviors that may lead to collisions, and ensure the safety and reliability of the robot's movement. Typical application scenarios include warehouse cargo handling (with dense stacking of goods), narrow channel navigation (such as pipeline inspection robots), and other tasks with high safety constraints.

(3) Complex dynamic scenarios: The DQN algorithm should be preferred. In such scenarios, the environment is dynamically changing (such as the movement of pedestrians, vehicles, or other obstacles), the state space is high-dimensional and uncertain, and the tabular algorithm is difficult to adapt due to the curse of dimensionality. The DQN algorithm can effectively fit the high-dimensional dynamic state through the deep neural network, has strong environmental generalization ability and robustness, and can maintain stable tracking performance in the face of unknown dynamic changes. Typical application scenarios include outdoor autonomous navigation of service robots, dynamic factory environment inspection, and other tasks that require strong adaptability to environmental changes.

4.2. Discussion and Limitations

This paper completes the comparative study of three basic reinforcement learning algorithms in the path tracking of two-wheeled mobile robots through simulation experiments, and obtains valuable algorithm selection suggestions. However, there are still certain limitations in the research, which need to be improved in future work: First, the simulation platform simplifies the actual physical environment and ignores non-ideal factors such as wheel slip, ground friction, and external interference. This makes the experimental results have a certain gap with the actual application scenarios, and the reliability of the algorithm in the real physical environment needs to be further verified through physical experiments. Second, the hyperparameters of the three algorithms in this study adopt a unified fixed setting. In fact, different algorithms may have optimal parameter combinations adapted to specific scenarios. The lack of personalized parameter tuning for different scenarios may lead to the failure to fully exert the performance potential of each algorithm. Third, this study only focuses on three basic reinforcement learning algorithms. With the development of reinforcement learning technology, advanced algorithms such as Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) have shown better performance in continuous control tasks[4]. The comparative analysis of these advanced algorithms in path tracking tasks needs to be further expanded.

Future research can focus on the following directions: First, optimize the simulation platform, introduce more real physical factors (such as variable friction coefficients, uneven ground, etc.), and build a simulation environment closer to the actual scene to improve the authenticity and reliability of the experimental results. Second, carry out research on adaptive hyperparameter optimization, design an adaptive parameter adjustment mechanism based on scenario characteristics, and further improve the performance of the algorithm in specific scenarios. Third, expand the scope of algorithm comparison, include advanced reinforcement learning algorithms, and conduct in-depth research on the performance differences and application scope of different types of reinforcement learning algorithms in path tracking tasks, so as to provide more abundant algorithm options for engineering practice.

5. Conclusion

This paper builds a two-wheeled mobile robot path tracking simulation platform based on Gym and Gazebo, and designs three map scenarios with different complexity levels (simple, complex, and dynamic obstacle scenarios). From the three core dimensions of convergence speed, control stability, and environmental robustness, a systematic comparative study is carried out on three basic reinforcement learning algorithms: Q-Learning, Sarsa, and DQN. The experimental results show that the three algorithms have obvious differences in adaptability to different scenarios: Q-Learning has the fastest convergence speed in simple static scenarios but insufficient robustness; Sarsa has excellent safe exploration ability and performs outstandingly in safety-priority complex static scenarios; DQN breaks through the dimensional limitation of tabular algorithms and shows significant adaptability and robustness in complex dynamic scenarios. The algorithm selection recommendations proposed in this paper for different application scenarios provide important theoretical reference and technical support for the engineering practice of two-wheeled mobile robot path tracking control. With the continuous development of reinforcement learning technology, combining scenario characteristics to carry out targeted algorithm selection and performance optimization will become an important direction to further improve the path tracking performance of mobile robots.

References

[1] Mnih, Volodymyr. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
[2] Rummery, Gavin A., and Mahesan Niranjan. On-line Q-learning using connectionist systems. Vol. 37. Cambridge, UK: University of Cambridge, Department of Engineering, 1994.
[3] Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).
[4] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).
[5] Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." Machine learning proceedings 1994. Morgan Kaufmann, 1994. 157-163.
[6] Busoniu, Lucian, et al. Reinforcement learning and dynamic programming using function approximators. CRC press, 2017.