Research on the Effect of Vehicle Obstacle Avoidance Problem Based on Q-learning

Wentao Kou *

Department of Mechanical Engineering, Taiyuan University of Technology, Taiyuan, 030024, China

* Corresponding Author Email: kouwentao8024@link.tyut.edu.cn

Abstract. Machine learning is one of the key topics for researchers. As an important branch of machine learning, reinforcement learning focuses on how agents learn strategies through trial and error in dynamic environments. The introduction of Q-learning was a major contribution to value-function-based methods in reinforcement learning. This passage uses a system of controlling variables for simulation and analysis based on Matlab Robot Playground to conduct research on the influence of parameters in Q-learning under different obstacle conditions and try to find the optimal choice of parameters in different situations. The two most important variables in this study are the learning rate α and the discount factor γ. When the alpha value is higher, the robot's path is simpler, and the learning takes a shorter time. The effect of the size of the gamma value on path planning is not regular. However, it can be found that the bigger the value is, the path choice of the agent will favor experience in the past. This study starts with the surface functions of the Q-learning algorithm and discusses the impact of major variables on the algorithm results, providing a unique approach for in-depth research on Q-learning algorithms.

Keywords: Machine learning; Q-learning; Robot; Matlab.

1. Introduction

The subfield of computing algorithms known as "machine learning" is a developing field of study that aims to emulate human intelligence by processing information from its surrounding environment. In the contemporary era of "big data," they are regarded as the workhorse [1]. With the advancement of technology, the path planning and navigation technology of robots has become more mature. However, as the environment becomes more complex, the limitations of traditional navigation methods are becoming increasingly prominent. To overcome this challenge, reinforcement learning was introduced in the path planning problem. As a significant branch of machine learning, Reinforcement Learning focuses on how agents learn policies through trial and error in dynamic environments. Q-learning is a general family of reinforcement learning techniques that are frequently used in artificial intelligence and statistics [2]. Q-learning is a simple way for agents to learn how to act optimally in controlled Markovian to find the optimal policy that maximizes the cumulative reward of an agent in an environment first introduced by computer scientist Christopher Watkins in 1989 [3]. The core idea of Q-learning is to learn a Q-value function that evaluates the value of taking a certain action in each state. In Q-learning, an agent needs to balance between exploring new actions and exploiting known optimal actions. This balance is a core challenge in reinforcement learning, maybe there is a better balance between these two parameters even more. Therefore, this passage is based on Matlab Robot Playground to put the effect of parameters in Q-learning on vehicle obstacles under the test of objective data. First, the research controls the learning rate and the discount factor separately so that they vary between 0 and 1 to observe their effect on agent learning. After that, observe the influence of another possible dependent variable. The core innovation of this article is the use of the Q-learning algorithm as a tool to solve the problem of car obstacle avoidance, while also analyzing the tool itself, or in other words, visualizing the learning results to more clearly express the impact of Q-learning on variables, laying a good foundation for further in-depth research in the future.

2. Materials and methods

2.1. Tools

In this paper, all studies are carried out based on Matlab, Matlab Robot Playground, and Matlab Robot Playground Expansion.

2.2. Model-free learning

Deep reinforcement learning (RL) has made significant progress in challenging decision-making scenarios on multiple occasions. However, these algorithms usually need a significant quantity of data before they operate at a reasonable level. When learning, performance can be dreadfully low. This could be suitable for a simulator, but it significantly restricts deep RL's application to numerous real-world activities, where the agent has to be taught in an authentic setting [4]. In the context of Markov Decision Processes (MDP), states are defined as abstractions of the surrounding environment, actions represent the available options for the agent, and for each state-action pair, and a distribution of potential subsequent states is established. Furthermore, a reward function is present, which allocates a random reward to every action and state [5].

Q-learning is a type of model-free reinforcement learning. Furthermore, it can be conceptualized as an asynchronous dynamic programming (DP) technique. Without having to create maps of the domains, it gives agents the ability to learn how to behave optimally in Markovian domains through the experience of actions. It can be demonstrated that, under the assumption that all actions are regularly sampled in all states and that the action values are represented discretely, Q-learning will converge to the optimal action values with probability [3].

In this virtual environment, the agent interacts with the environment and gathers a numerical reward that represents performance, thereby training a model through reinforcement learning methods. The agent begins each time step in the state S_t and executes an action to move into a new state, S_{t+1}, to get a reward, r_{t+1}

The policy $\pi(a \mid s)$ represents a component of the agent that selects the action to be executed in a given state. Consequently, it is a key element that requires training.
The return is a cumulative reward, γ being a discount factor.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3\ldots} \tag{1}$$

The value function represents the anticipated reward associated with the implementation of a specified policy, contingent on the initial state of the system.

$$V_\pi(s) = E[G_t \mid S_t = s] \tag{2}$$

The Q-function, or state-action value function, represents the anticipated return from a specific initial state and action.

$$Q_\pi(s,a) = E[G_t \mid S_t = s, A_t = a] \tag{3}$$

The probability is$P^a_{ss'}$, and the state transition to a state S' depends simply on the current state s and action a.

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V_\pi(s') \tag{4}$$

$$Q_\pi(s,a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma \sum_{a'} \pi(a' \mid s') Q_\pi(s'a,a') \tag{5}$$

Q-learning strikes a balance between exploration (the undertaking of novel actions) and exploitation (the selection of the most optimal actions that have been previously identified). One of the most frequently employed approaches is the ε-greedy policy, which entails selecting the action with the highest current Q-value with a probability of 1-ε and a random action with a probability of ε.

2.3. Grid world

This passage examines the projective simulation model, a cutting-edge form of artificial intelligence. It was recently demonstrated that the PS agent outperforms established models of reinforcement learning in several straightforward task contexts [6].

In the following experiment, first use reinforcement learning to determine the barrier-free path between two waypoints, then create a 2D environment using the following methods:

State: In the grid universe, each cell is a state.

Actions: The agent has four options in each cell: left, right, up, or down.

Reward: The agent receives a positive reward if the goal is reached; a negative reward if a barrier or world boundary is encountered; and no reward if the goal is not reached.

This study uses the real-time scripts in Matlab Robot Playground Expansion Unit 6 for model simulation of Q-learning setting up the components in the model, changing the code to generate a different obstacle map. The size of the maps is 20×20 cells and 30×30 cells.

Q-learning employs an off-policy technique to distinguish between the acting policy and the learning policy, in contrast to earlier algorithms that did not differentiate between behavior and learning. The difficulty is that it is a poor option because the information was not incorporated in the update of the Q-function of the current state, even if the action chosen in the following state was mediocre [7]. However, the problem is resolved by Q-learning because it uses off-policy. The following is the formula for the Q-value [8].

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma max - Q(s,a)] \tag{6}$$

Where α is the learning rate. Then initialise the Q-table with:

Q = zeros (WorldX, WorldY, 4);

And set the parameters for the Q-function Alpha and Gamma.

2.4. Agent training

The Q-learning agent will be trained by repeatedly simulating its behavior, collecting rewards, and updating the Q-function. At the same time, set Episodes and maxNumsteps for the agent.

The action selection function chooses to move to the next state with the maximum value of Q. If one or more actions have equal values of Q, the action is chosen at random.

2.5. System of controlling variables

In the process of Q-Learning, each parameter affects the navigation performance of the agent in the final iteration. Here are four main parameters, α is the learning rate, γ is the discount factor, Episodes is the number of learning Steps, and maxNumsteps is the maximum number of times that an agent can perform in one Episode. Using the controlling variables system to control each parameter then observe the result of learning.

Because the learning result of the agent is random, under the same conditions, each Episode used to complete the learning is also different, so the data corresponding to each point in the line chart given in this paper is taken from the average of the five results.

2.6. Hypothesis

Before starting the simulation, this experiment would like to make two assumptions first. For the Q-learning algorithm, this paper proposes two hypotheses to capture the effect of the parameters on the results.

The larger the alpha value, the less time is needed to plan the avoidance path.

The lower value of gamma has less effect on the results of path planning for obstacle avoidance.

3. Results

3.1. The Impact of Learning Rate Alpha

The two most important parameters in this experiment are α and γ, so this paper explores their influence first.

In the experiment, the three variables gamma, Episodes, and maxNumsteps were unchanged and kept at a moderate value. This study averaged the five results from each parameter value run out and represented them in a line graph. These two line graphs result in data showing that the higher the alpha value is, the learning time and Episodes show a decreasing trend. After observing the path result shown in Fig. 1. Which is generated by the path generation function, it can also be found that when the alpha value increases, the learning result path of the robot is simpler, the less step is used, and the path tends to the shortest path. On the contrary, when the alpha value is smaller, the path diverges, and the end of the learning result is farther away from the target location.

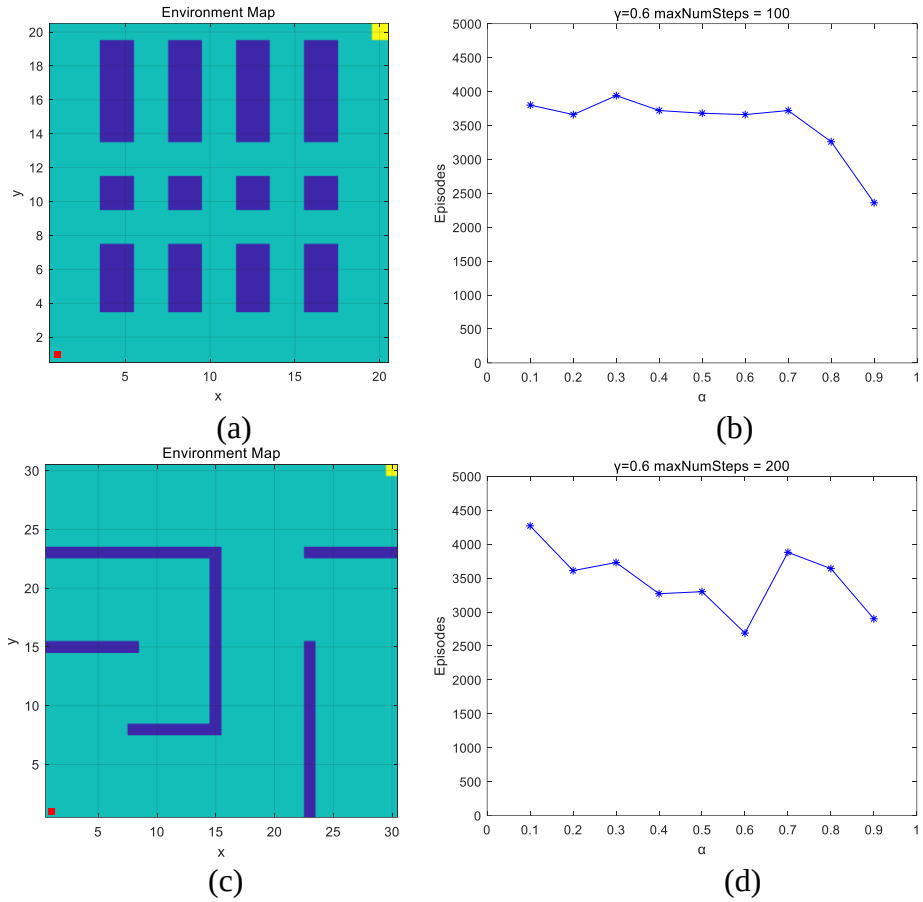


Fig. 1 The effect of Alpha in two obstacle maps. (a) Obstacle Map 1; (b)The impact of α on map 1; (c) Obstacle Map 2; (d)The impact of α on May 2. (Photo/Picture credit: Original)

3.2. The impact of the discount factor

In the experiment, control the three variables alpha, Episodes, and maxNumsteps unchanged and kept them at a moderate value. For gamma, the effect of changes on the path planning results is oscillatory, and different degrees of up and down fluctuations can be seen in the curves of each map. The reason for this situation was analyzed is because gamma represents how much is learned from previous experience; this experience is not always correct, while each calculation is different and cannot be anticipated. There exist two meanings for the discount factor $\gamma \in [0, 1)$. First, it can be viewed as a characteristic of the problem that needs to be handled, giving priority to immediate gains above long-term gains. Secondly, in activities that are not episodic, the discount factor ensures that each action value is well-defined and has a finite value [9]. The asymptotic rate of convergence for

Q-learning has been determined under the assumption that the next state distribution is identical for each state. This finding demonstrates the exponential dependence of the asymptotic convergence rate of Q-learning on 1−γ [10]. The impact of gamma variation on learning Episodes is shown in Fig. 2.

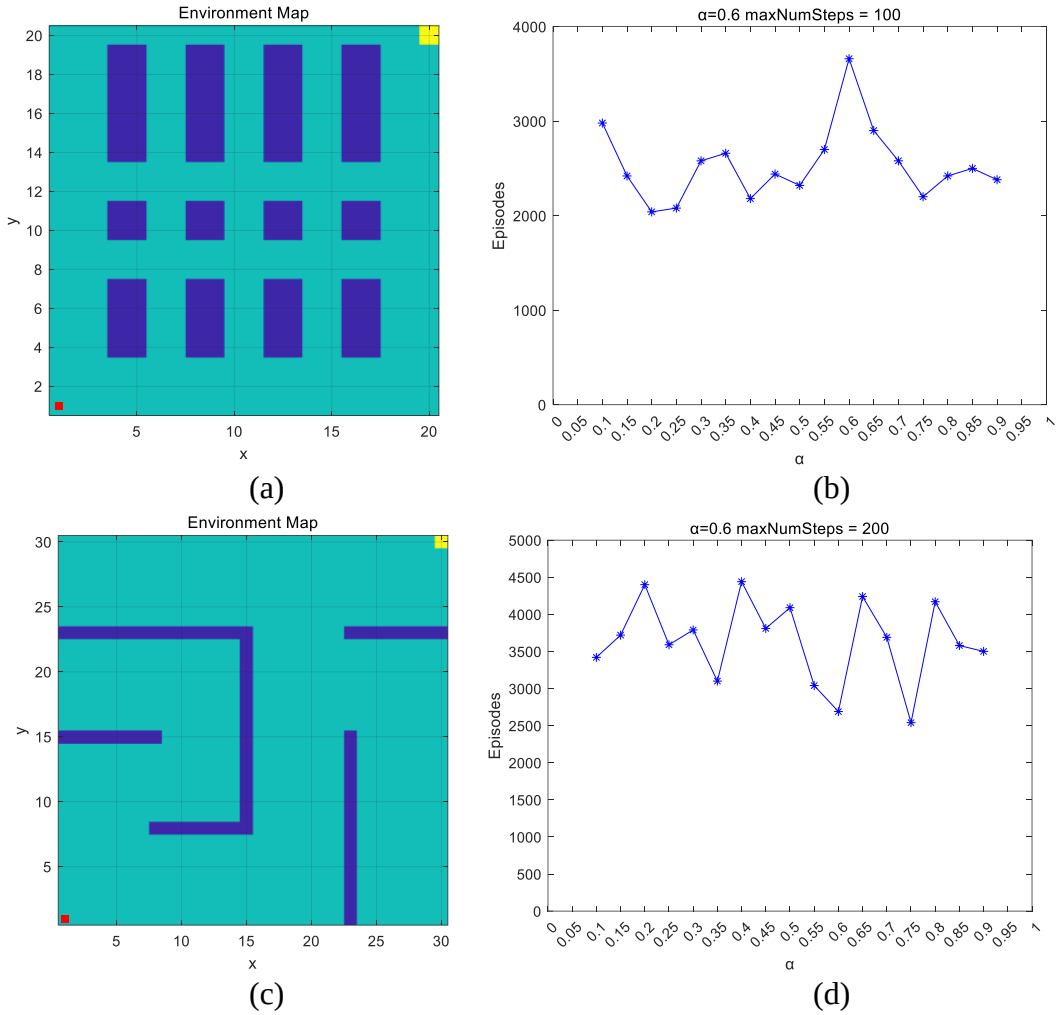


Fig 2. The effect of Gamma in two obstacle maps. (a) Obstacle Map 1; (b)The impact of γ on map 1; (c) Obstacle Map 2; (d)The impact of γ on map 2. (Photo/Picture credit: Original)

3.3. The impact of Episodes and maxNumsteps

In this part, control alpha and gamma unchanged during the simulation, and observe how the agent learns by Episodes and maxNumsteps.

First of all, control Episodes equal to 15,000, change the maxNumsteps from 100 to 200, in this process, the number of times the agent learns has significantly decreased, because in each learning session, the number of times the agent can choose a state has increased. With each learning session, it can approach the goal more closely. The effect of maxNumsteps is shown in Fig. 3.

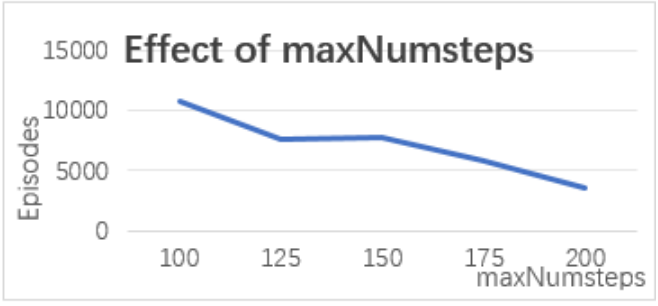


Fig 3. The effect of maxNumsteps in Obstacle Map 2. (Photo/Picture credit: Original).

Second, continue to control other variables unchanged in the experiment and change the Episode. The results demonstrate that, under the assumption that all actions are regularly sampled in all states and that the action values are represented in a discrete manner, Q-learning converges to the optimal action values with probability 1. Additionally, it proposes expansions to non-discounted but absorbing Markov environments and scenarios in which many Q values, as opposed to just one, can be modified per iteration [3].

In fact, in the previous experiments and preparation process, the influence of maxNumsteps has already been obvious. As can be seen from the previous text, learning results will eventually converge to 1. Therefore, at present, if Episodes do the hookup, the agent will reach the goal. However, due to cost and time considerations, controlling the Episodes within a certain range is beneficial for the efficient completion of the experiment.

After that, this research also observed the impact of changes in alpha and gamma on the number of steps used for agent learning results. The effect is shown in Fig. 4.

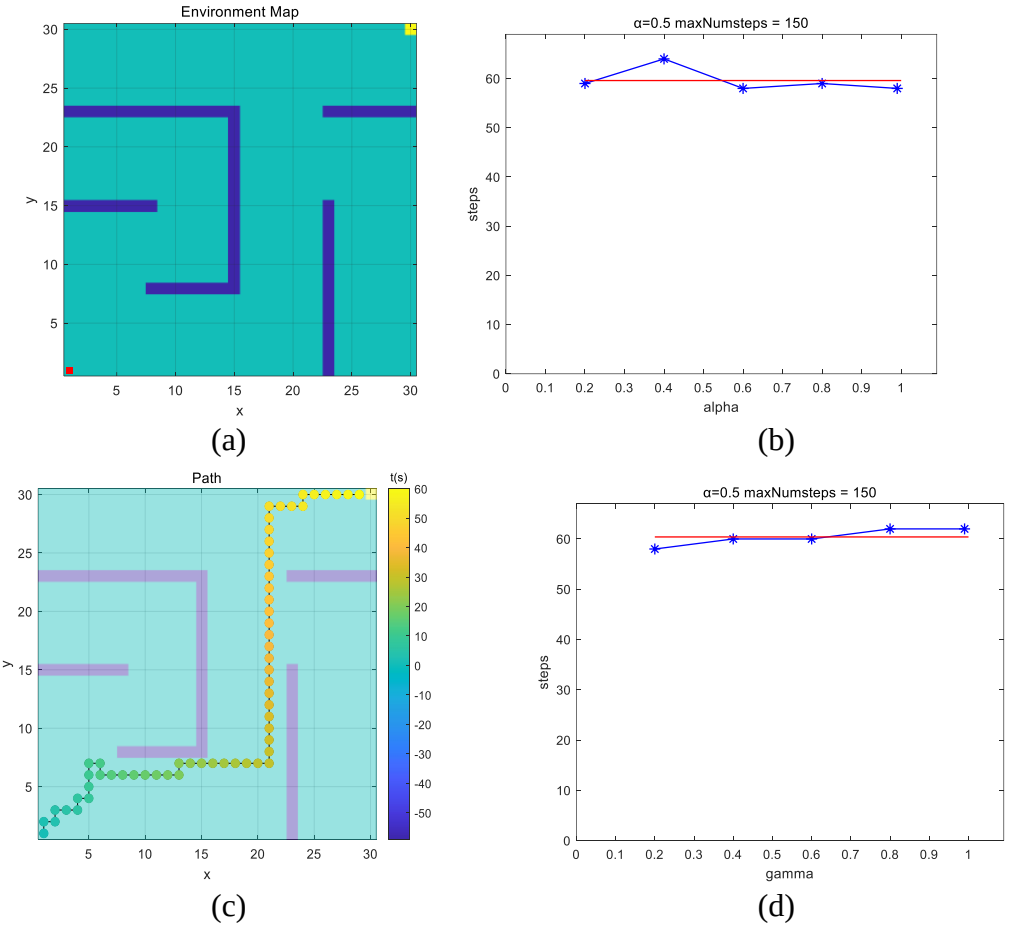


Fig 4. The effect of Alpha and Gamma on a number of steps in Obstacle Map 2. (a) Obstacle Map 2; (b)The impact of α in the step of learning result; (c)Result path of Obstacle Map 2; (d)The impact of γ in the step of learning result. (Photo/Picture credit: Original).

This paper counted the number of steps used each time in path planning according to different parameter values; to observe the effect of alpha and gamma on the number of steps. The effect of both parameters on the number of steps is minimal, and within an error of no more than 7.1%, it can be said that the number of steps corresponding to the different parameter values in each map does not change.

The reason for doing this is that in different situations, the goal is to find the optimal path, not the shortest path. Therefore, when the agent considers the path feasible during the learning process, it often does not make other attempts in subsequent learning, so the impact on the number of steps is not significant. This is also the reason why redundant routes in the presented result path graph.

3.4. Comparison with hypothesis

Simulation experiments show that the larger the alpha value, the less time is needed to plan the avoidance path. The lower value of gamma has less effect on the results of path planning for obstacle avoidance. So the assumptions made earlier were roughly correct.

4. Discussion and Analysis

4.1. Reasons for the formation of research results

Based on this work, the passage analyses the impact on the planning of these parameters. Each result completes the obstacle avoidance problem. But it still needs to focus on their influences. Given that the reasons for the impact of Episodes and numMaxsteps on agent Q-learning outcomes have been explained in the previous text, the discussion summarizes the reason for alpha and gamma.

The higher the alpha value is, the faster the iteration. Based on this point, the alpha, which is also the learning rate, represents how much the new Q value learns from the old Q value as a proportion of itself. The smaller the alpha value, the paths diverge and the iteration slower is. The change in alpha does not affect the number of path steps.

As to the reasons for these situations, this paper summarizes two points for alpha:

1) Alpha is present in both components of the Q-value, and so can greatly influence decision-making.

2) Alpha represents how much has been learned from the old Q-value, which is the optimal solution in the previous state, and the more it learns from it, the more efficient the algorithm will be.

$$\text{new } Q_{S,A} = (1-\alpha)Q_{S,A} + \alpha(R_{S,A} + \gamma * \max Q'(s', a')) \quad (7)$$

The smaller the gamma is, the less perceptive the path is to the environment and there will be more forks. Changes in the gamma value have an oscillatory effect on the algorithm time. The change in alpha does not affect the number of path steps.

As to the reasons for these situations, this paper summarizes two points for gamma:

1) The larger the value is, the greater the effect of maxQ^' (s^', a^'), which is unpredictable and varies from time to time.

2) The larger the value is, the robot will learn according to the previous 'experience' of getting a reward, which may not be correct. This means that it takes more time for the vehicle to return to its original route.

From the above two points, it is understandable why the effect of gamma is fluctuating.

4.2. Limitations analysis

This passage is only based on the simulation environment in Matlab for research, which has certain applicability issues. The learning behavior of agents also tends toward primitive machine learning. During the research process, it was found that when the starting point of the agent is in the left half of the map and the target is in the right half, the agent almost does not attempt to move to the left. Therefore, in complex terrain, the agent is often trapped at obstacles that require a U-turn. The research on Q-learning in this article is only limited to the surface level of algorithms, and there are still many deeper issues worth studying. For example, researchers can propose a framework for modeling Q-learning agent behavior using ε-greedy exploration mechanisms, the objective is to analyse the continuous-time versions of Q-learning update rules and investigate the impact of the presence of other agents and ε-greedy mechanisms on them [11].

5. Conclusion

This article is based on the core method of controlling Q-learning parameter changes to study the impact of various variables in Q-learning on the results of agent learning. This research has shown

that in the process of using Q-learning to solve the Vehicle Obstacle Avoidance Problem, the variable Alpha controls the update speed and determines the impact of new information on Q-value updates. The range of learning rate α is between 0 and 1. When α is large, the weight of new information is higher and the Q value is updated faster. Secondly, the value of α affects the balance between new and old information. A higher Alpha allows the Q value to quickly reflect the latest observation results, which is suitable for dynamic environments or situations that require quick adaptation.

In Q-learning algorithms, Gamma is a discount factor that plays a crucial role in controlling the value of future rewards. Gamma determines the degree to which the agent values future rewards. When γ approaches 1, the agent considers long-term rewards to be almost as important as immediate rewards because future rewards have fewer discounts. Adjusting gamma can also balance short-term and long-term returns. A smaller Gamma value reduces the impact of future rewards, and agents will prioritize short-term gains.

Therefore, it is recommended that researchers choose larger Alpha values and moderate Gamma values in the process of using Q-learning to improve learning efficiency.

Starting from the surface functions of the Q-learning algorithm, this article analyzes the impact of variable changes on the results. Future research can delve deeper into the logic of the algorithm from the perspective of a program or code. For example, researchers can propose a framework for modeling Q-learning agent behavior using ε-greedy exploration mechanisms, analyze continuous-time versions of Q-learning update rules, and investigate how the presence of other agents and ε-greedy mechanisms affect it.

References

[1] El Naqa I, Murphy M J. What is machine learning? Springer International Publishing, 2015.

[2] Clifton J, Laber E. Q-learning: Theory and applications. Annual Review of Statistics and Its Application, 2020, 7(1): 279-301.

[3] Watkins C J C H, Dayan P. Q-learning. Machine Learning, 1992, 8: 279-292.

[4] Hester T, Vecerik M, Pietquin O, et al. Deep Q-learning from demonstrations. Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[5] Even-Dar E, Mansour Y, Bartlett P. Learning Rates for Q-learning. Journal of Machine Learning Research, 2003, 5(1),

[6] Melnikov A A, Makmal A, Briegel H J. Projective simulation applied to the grid-world and the mountain-car problem. Preprint arXiv: 1405.5459, 2014.

[7] Dietterich T G. Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 2000, 13: 227-303.

[8] Jang B, Kim M, Harerimana G, et al. Q-learning algorithms: A comprehensive classification and applications. IEEE Access, 2019, 7: 133653-133667.

[9] Hasselt H. Double Q-learning. Advances in Neural Information Processing Systems, Proceedings. Neurips. Cc, 2010, 23.

[10] Azar M G, Munos R, Ghavamzadeh M, et al. Speedy Q-learning. Advances in Neural Information Processing Systems. 2011.

[11] Rodrigues Gomes E, Kowalczyk R. Dynamic analysis of multiagent Q-learning with ε-greedy exploration. Proceedings of the 26th Annual International Conference on Machine Learning. 2009: 369-376.