Deep Q network algorithm based on sample screening

Hongbo Zhang, Peng Wang*, Cui Ni, Nuo Cheng
School of Information Science and Electrical Engineering, ShanDong Jiao Tong University, Jinan, Shandong, China

ABSTRACT

Path planning acts a significant role in the motion and exploration of mobile robots. As artificial intelligence develops, path planning is also moving towards intelligent direction. Deep Q Network (DQN) has low computational complexity and high flexibility, and is widely used in mobile robot path planning. DQN algorithm mainly obtains training sample data through uniform random sampling, which is easy to generate redundancy and reduce the precision of the training model. So as to reduce the redundancy of selected samples, an improved DQN method for path planning is proposed in this paper. By establishing sample similarity screening matrix, the proposed algorithm can eliminate samples with high similarity, improve model training effect, and further enhance the precision of path planning. Simulation results show that the algorithm this paper proposed has a great improvement in the convergence speed of DQN model training and the robustness of path planning.

Keywords: Path planning, DQN, similarity, samples

1. INTRODUCTION

As robot technology[1] develops, mobile robot is more and more widely used. As the guarantee of robot movement, path planning is especially important. The goal of path planning[2-3] is finding a safe way that keeps from collision from jumping-off point to the target point in specified areas. Traditional path planning algorithms, such as artificial potential field method[4], Dijkstra algorithm[5] and rapidly-exploring random trees (RRT) method[6], are easy to implement to finish the path planning in known environment, but there is a problem of poor exploration ability in path planning. In this case, there will be more space of reinforcement learning.

Reinforcement Learning (RL)[7-8] is an artificial intelligence algorithm that does not require prior knowledge and obtains feedback information through trial-and-error iteration with the environment to optimize strategies. As one of the methods of machine learning[9], it is suitable for solving sequential decision problems. In manner of trial and error in the environment, the agent takes the maximum cumulative reward brought by doing something as its optimal learning strategy. Markov Decision Process (MDP) is used to describe reinforcement learning problem mathematically. In the whole process of reinforcement learning exploration, for each time step, the environment provides an observation state S_t for the agent, and the agent selects an action A_t according to the strategy to respond, and then gets the immediate reward R_t provided by the environment and moves to the next state S_{t+1}. The agent's goal is to maximize the expected discount reward by looking for an optimal strategy.

On the basis of reinforcement learning, deep reinforcement learning increases the perceptual ability of deep learning. As one of the representatives of deep reinforcement learning, DQN algorithm[10-12] has not only been tested with good effects in Atari games, but also has excellent performance in path planning. Based on Q-learning algorithm, DQN algorithm replaces Q-table with convolutional neural network, which solves the problem of large state space. DQN breaks the correlation between data by random sampling, and uses both the target network and the current network to solve the problem of model stability. Compared with other models, DQN has strong versatility and can be used to solve different types of problems.

*205049@sdjtu.edu.cn

International Conference on Computer Application and Information Security (ICCAIS 2021), edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 122600I · © The Authors. Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.2637373

2. IMPROVED ALGORITHM

2.1 Improvement of training data

DQN algorithm combines the powerful representation ability of deep neural network with Q-learning algorithm, and achieves excellent performance in a series of tasks. The success of this algorithm benefits from the target network and experience replay method used in DQN algorithm, especially experience replay method, which stores previous experiences in a container and disrupt correlations between them. DQN algorithm uses experience replay method to reduce the correlation between training experiences, improve experience utilization and reduce overfitting. The main method is to store the experience generated in the process of finding the target into the experience pool. During network training, a batch of experience is sampled without order or plan from the experience pool for updating network parameters. However, these experiences are not equally important for network model learning, and there will be repeated and similar experiences. To solve this problem, an improved experience replay method is proposed to reduce similarity, which uses an experience pool to store all experiences, and speeds up network model training by eliminating similar experiences in the training process. That is, when a batch of sample data is extracted each time, the matrix as shown in Figure 1 is designed for similarity screening.

1	2	4
128	0	8
64	32	16

Figure 1. Screening matrix.

1	0	1
0	0	0
0	1	0

Figure 2. The matrix that describes the surroundings.

While collecting samples for training, the surrounding conditions of the current point are calculated and a 3×3 matrix representing the surrounding obstacles is returned, as shown in Figure 2.

Take the Hadamard product of the two matrices and sum them to get a number. Through experimental verification, when the matrix representing the surrounding obstacles changes, the final result will also change, which indicates that the designed matrix can be used to calculate the similarity of training samples. Through this method, the sample data with high similarity is removed, redundant samples are reduced, and the diversity of samples is retained. This is of great help to the training of agent's exploration ability.

2.2 Reward function settings

Deep reinforcement learning emphasizes the interaction between agent and environment, adopts the "interaction-trial-error" mechanism, and maximizes the cumulative reward through the interaction updating strategy between agent and environment. Reward is the key for agent to realize strategy update. Improper reward setting or unclear feedback will affect agent's training effect.

According to the principle that the shorter the path, the greater the cumulative reward, this paper also improved the reward function in the original DQN algorithm, as shown in equation (1).

$$R=\begin{cases} +1 & \text{Reach target point} \\ -1 & \text{Hit an obstacle} \\ \max\left(-1,\ 1-\dfrac{a+b}{c}\right) & \text{Other} \end{cases} \tag{1}$$

where a is the length from the position of the agent to the starting point, b is the length from the position of the agent to the target point, and c is the length from the starting point to the target point.

The reason why the reward function is designed in this way is that when the agent takes the action, the length from the position of the agent to the target point will also change. The closer the distance is, the bigger the reward will be, and the farther the distance is, the smaller the reward will be. In this way, the training efficiency of the model is improved and convergence speed is accelerated.

The proposed algorithm flow is as follows.

Step 1: Initialize Replay Buffer D with capacity N, state value model Q and parameter θ, Target Network Q^{-} and parameter θ^{-}, and set learning rate α, discount factor γ, degree of exploration ε and other parameters.

Step 2: Initialize training environment of the agent, get the initial state s_t, explore randomly and select an action a_t with the probability of ε, or select the current optimal action with the probability of $1\text{-}\varepsilon$, $a_t = \max_a Q^*(\phi(s_t), a; \theta)$.

Step 3: Perform the action a_t to get a new state s_{t+1} and reward r_t, save $\{s_t, a_t, r_t, s_{t+1}\}$ in D.

Step 4: Collect training samples from D, remove samples with high similarity, and then train and calculate,

$$y_j = \begin{cases} r_{j+1} & \text{if terminal} \\ r_{j+1} + \gamma \max_{a'} Q(\varphi_{j+1}; a'; \theta^{-}) & \text{if non-terminal} \end{cases}$$

Step 5: The stochastic gradient descent method is used to calculate $loss = (y_j - Q(\phi_j, a_j; \theta))^2$

Step 6: Repeat Steps 2 to 5 to update the Target Network after M steps.

3. EXPERIMENTAL RESULTS AND ANALYSIS

This paper uses PyCharm development platform and TensorFlow1.x framework for programming. For the sake of illustrate sufficiently that improved algorithm is better than the original algorithm, the training process and training effect are compared.

The improved algorithm eliminates the data samples with high similarity, reduces the similarity of data samples, decreases the redundancy of sample and increases the convergence speed of training. The dashed line represents the result of the original DQN algorithm, the solid line represents the result of the improved algorithm. They are shown in Figure 3.

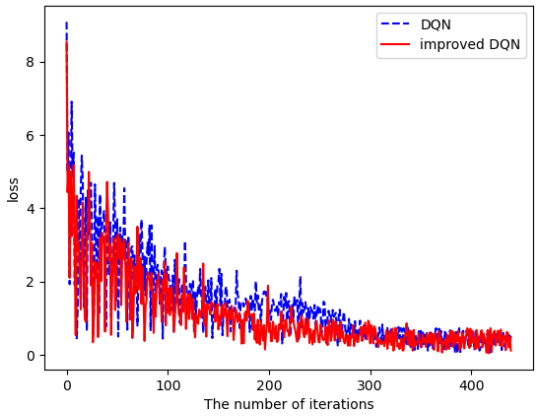

Figure 3. Comparison of loss values between our improved DQN and original DQN.

As can be seen from Figure 3, both the DQN algorithm and the improved DQN algorithm have large fluctuation of loss curves at the beginning. DQN algorithm gradually flattens after 300 rounds of training, while the improved DQN algorithm has been flattening after about 200 rounds of training. Obviously, the improved DQN algorithm converges faster.

So as to fully demonstrate the performance improvement of the improved DQN algorithm on path planning, complex environment and simple environment are set up in this paper. In each environment, 20 randomly generated maps are used to test the DQN algorithm and the improved DQN algorithm respectively. Initially, the agent starts from the blue grid in the upper left corner, its destination is the green grid in the lower right corner, and the white grids are the route where the Agent can walk. In the map, 75 black squares representing obstacles are randomly generated. If the agent successfully

reaches the target point without encountering obstacles, the path found is drawn. Otherwise, it will fail and the image will not be displayed.

Figures 4 and 5 show the test map and test results of DQN algorithm respectively.

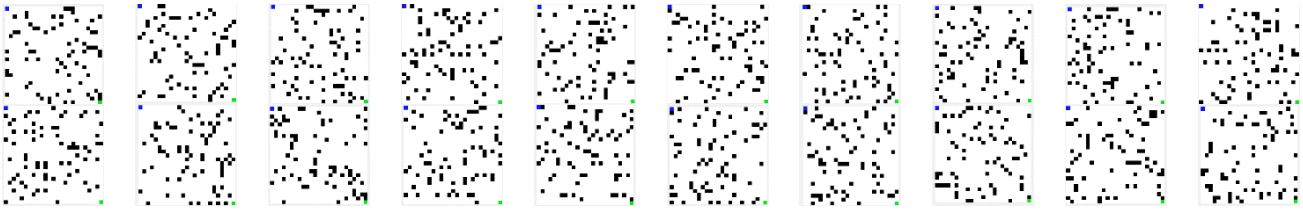

Figure 4. Test map of DQN algorithm.

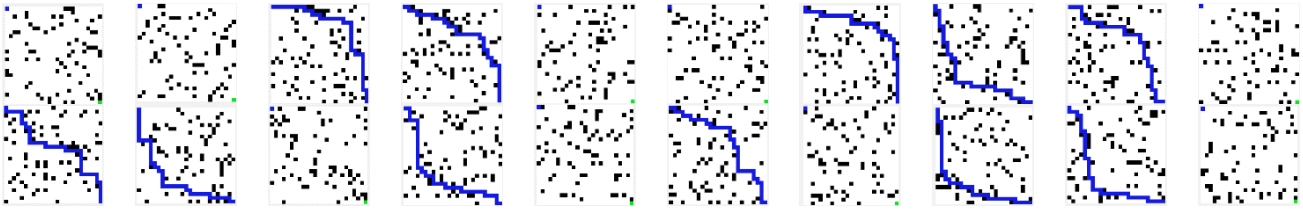

Figure 5. Test results of DQN algorithm.

DQN algorithm is tested by using grid map of the same size. After training, its ability of path planning has been greatly improved. Of the 20 test maps, 11 agents successfully reached the target point, with success rate of 55%.

Figures 6 and 7 show the test map and test results of the improved DQN algorithm respectively.

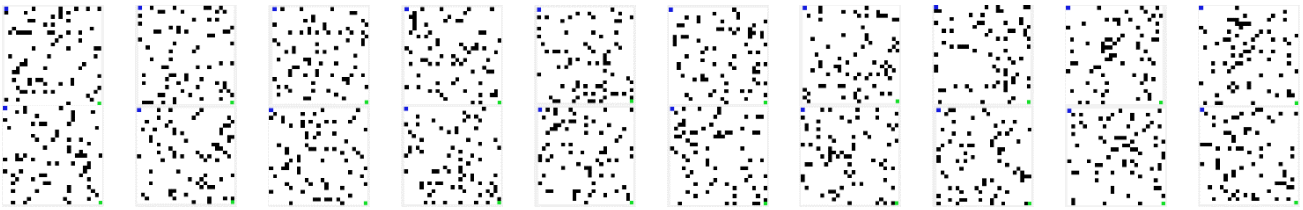

Figure 6. Test map of improved DQN algorithm.

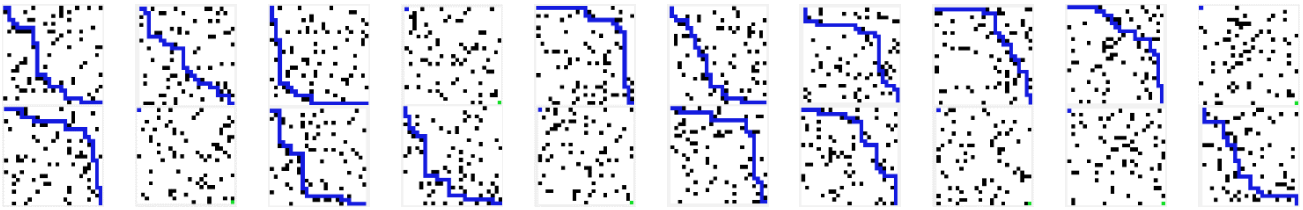

Figure 7. Test results of improved DQN algorithm.

The algorithm in this paper was tested on 20 maps, and 14 agents successfully reached the target point, with success rate of 70%. In the training stage, the algorithm in this paper eliminated the data samples with high similarity, increased the diversity of training samples, and the agent learned more “ability” to deal with complex scenes. The success rate of this algorithm in complex scenes is 15% higher than that of DQN algorithm.

Next, the DQN algorithm and the improved DQN algorithm are tested in a simple environment. The number of obstacles in the simple environment has been reduced to 40. Figures 8 and 9 show the test map and test results of DQN algorithm respectively.

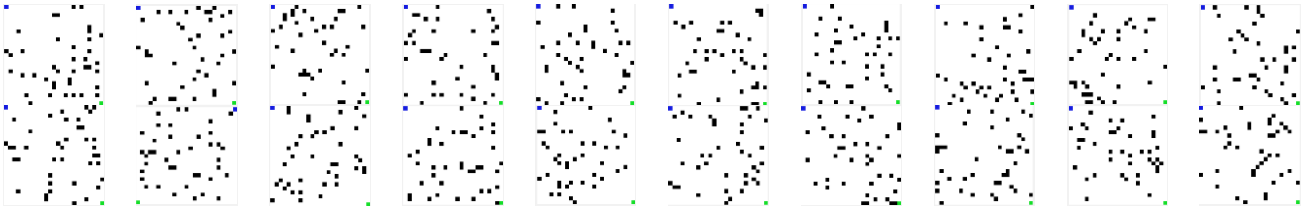

Figure 8. Test map of DQN algorithm.

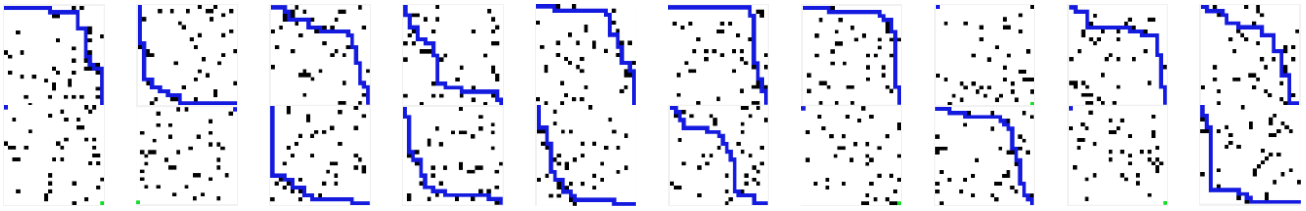

Figure 9. Test results of DQN algorithm.

It is found that the success rate of DQN algorithm in simple map is higher than that in complex environment, reaching 75%.

The improved DQN algorithm was tested on grid maps of the same specification. Figures 10 and 11 show the test map and test results of the improved DQN algorithm respectively.

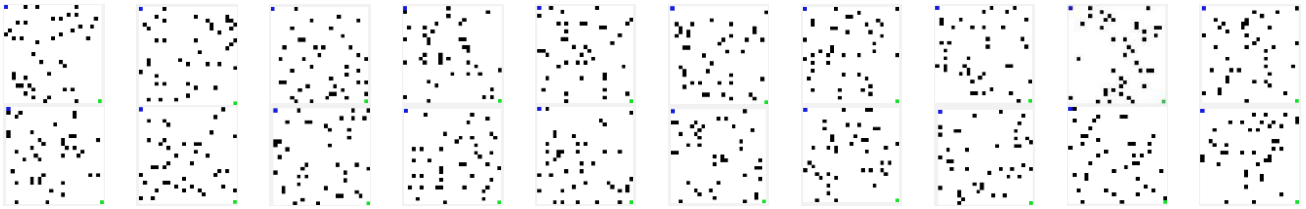

Figure 10. Test map of improved DQN algorithm.

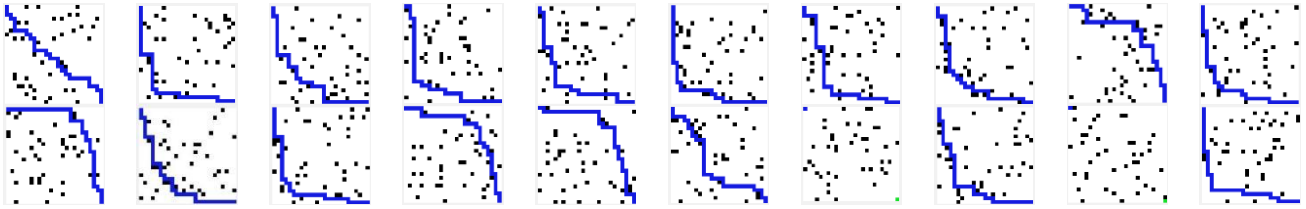

Figure 11. Test results of improved DQN algorithm.

Both the DQN algorithm and the improved algorithm achieve good results in simple environment. The success rate of DQN algorithm for path planning is 75%, and the success rate of the improved DQN algorithm is 90%. The success rate of the improved DQN algorithm is 15% higher than the original DQN algorithm.

4. CONCLUSION

Aiming at the existing problems of low sample training efficiency and sample redundancy of the original DQN algorithm, sample similarity screening matrix is used in this paper to solve them. The algorithm in this paper optimizes the training mode of data samples by using the designed matrix, eliminates the data samples with high similarity, obtains high-quality sample data, and improves the accuracy of the training model. In the training process, let more useful training samples be used. In addition, the proposed algorithm optimizes the reward function of the original DQN algorithm. The reward function designed in this paper comprehensively considers obstacles and the distance between agent and target point. Agent can plan a closer path while avoiding obstacles.

ACKNOWLEDGMENTS

This work was partially supported by Shandong Provincial Transportation Science and Technology Project (Grant No. 2021B120).

REFERENCES

[1] Blanco, C., "Artificial intelligence," Cadmus, 4, 79 (2021).
[2] Wang, H. and Pan, W. J., "Research on UAV path planning algorithms," IOP Conference Series: Earth and Environmental Science, 012120 (2021).
[3] Kaushlendra, S. and Rajesh, D., "Path planning for robots: an elucidating draft," Journal of Control Science and Engineering, 4, 294-307 (2020).
[4] Liu, X. J. and Dou, Y. K., "Research on obstacle avoidance of small cruise vehicle based on improved artificial potential field method," Journal of Physics: Conference Series, 012036 (2021).
[5] Luo, M., Hou, X. R. and Yang, J., "Surface optimal path planning using an extended Dijkstra algorithm," IEEE Access, 8, 147827-147838 (2020).
[6] Zhou, Y., Zhang, E. D., Guo, H. L., Fang, Y. H. and Li, H., "Lifting path planning of mobile cranes based on an improved RRT algorithm," Advanced Engineering Informatics, 50, 101376 (2021).
[7] Yuan, Y. M., Li, H. Y. and Ji, L. L., "Application of deep reinforcement learning algorithm in uncertain logistics transportation scheduling," Computational Intelligence and Neuroscience, 5672227 (2021).
[8] Wang, X. S., Gu, Y., Cheng, Y. H., Liu, A. P. and Chen, C. L. P., "Approximate policy-based accelerated deep reinforcement learning," IEEE transactions on Neural Networks and Learning Systems, 31, 1820-1830 (2020).
[9] Mrinal, R. B. and Subhedar, J. M., "Autonomous driving architectures: Insights of machine learning and deep learning algorithms," Machine Learning with Applications, 6, 100164 (2021).
[10] Sun, Y. X., Yuan, B., Zhang, T., Tang, B. J., Zheng, W. W. and Zhou, X. Z., "Research and implementation of intelligent decision based on a priori knowledge and DQN algorithms in wargame environment," Electronics, 9, 1668 (2020).
[11] Chu, Z. Z., Sun, B., Zhu, D., Zhang, M. J. and Luo, C. H. M., "Motion control of unmanned underwater vehicles via deep imitation reinforcement learning algorithm," IET Intelligent Transport Systems, 14, 764-774 (2020).
[12] Yu, J. L., Su, Y. C. H. and Liao, Y. F., "The path planning of mobile robot by neural networks and hierarchical reinforcement learning," Frontiers in Neurorobotics, 14, 6 (2020).