ISSN: 2664-9640

Autonomous control decision algorithm based on improved deep reinforcement learning

Hang Yan, Chengliang Yan, Luqi Han

Chengdu University of Technology, China

Abstract

With the rise of neural networks, reinforcement learning has performed better and better in many traditional games. However, these performances cannot be applied to autonomous driving, because the state space in the real world is extremely complex, and the action space is continuous, requiring fine control. In order to ensure the stability of autonomous driving in complex environments, the Deep Deterministed Policy Gradient (DDPG) algorithm is selected to replace the traditional control method. This algorithm has good ability to deal with continuous control problems in complex environments. Gazebo is selected as the simulation environment, and quantitative and qualitative conclusions are given.

Keywords

Deep reinforcement learning; autonomous driving; gazebo; continuous control.

1. Introduction

Autonomous driving is an important research field in computer vision and control systems. In addition, deep reinforcement learning technology has been applied to various games and has achieved great success. The successful practice of deep reinforcement learning algorithms shows, Can use computer vision as input to direct high-dimensional state and action space to the agent to solve the control problem of complex environment. However, the current success of deep reinforcement learning is limited to discrete control functions. For autonomous driving, The input image contains complex backgrounds and objects, which involve many difficult visual tasks, such as target detection, scene understanding, etc. The controller needs to act correctly and quickly in this situation and reach the end point as quickly as possible under safe conditions. To this end, this article uses the DDPG algorithm. DDPG algorithm is a subject that combines DPG (Deterministed Policy Gradient) algorithm and deep reinforcement learning. This simulation experiment was carried out on the Gazebo platform, and finally analyzed the results to verify that the proposed model is effective.

2. Research status

Mobile robot navigation is a continuous decision-making process. The robot needs to perform different actions to avoid local obstacles or move towards the target point according to different sensors to perceive changes in the environment and its own state. Based on reinforcement learning, map-free navigation sensors that perceive the surrounding environment mainly include lidar, depth camera and monocular camera.

Among them, the lidar often uses the sparse distance value. For example, Lei T et al. use the sparse lidar signal as the sensing input, and use the asynchronous improved version of DDPG (Asynchronous DDPG) algorithm to train the robot to reach the target point [1]. The depth camera can directly obtain the distance value from the obstacle. Tail L et al. used the depth map as the visual input, and used GAIL to realize the dynamic ritual avoiding people [2]. The RGB image collected by the monocular camera contains richer environmental information. At the

ISSN: 2664-9640

same time, because it cannot directly obtain the distance information from the obstacle, a more complex navigation strategy is required. Lei T et al. [3] trained an exploration strategy with DQN In order to control the robot to continue to move and avoid collisions in the process; Zhu Y et al. [4] simultaneously input the image currently perceived by the robot and the target image to the twin network composed of the pre-trained ResNet-50 and the fully connected layer on ImageNet (In the Actor-critic network composed of Siamese Network) and the fully connected layer, output values and visual navigation strategies are used to solve visual navigation in small environments such as indoors; Mirowski P et al. respectively proposed a Nav A3C network with a multilayer LSTM structure combined with loopback Detection and depth prediction are used to achieve navigation tasks in complex environments [5] and MultiCity Nav network structure to achieve city-scale navigation [15].

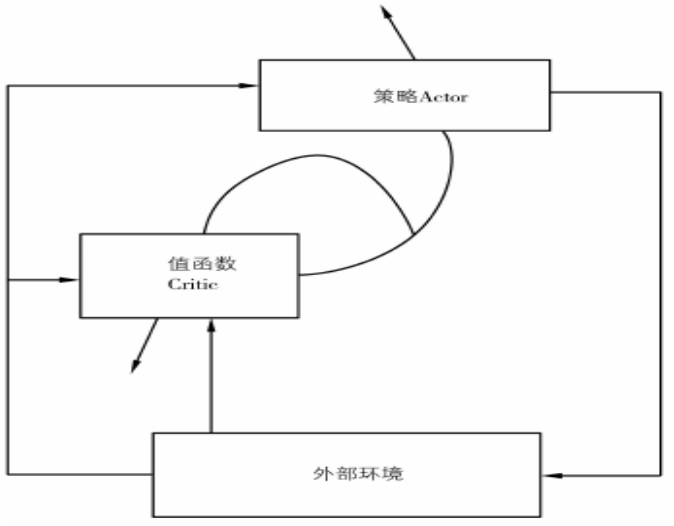
3. Algorithm

In autonomous driving, if the value-based learning method is used, the results will be worse, and the strategy-based method can solve this problem well. Strategy-based methods are divided into random strategies and deterministic strategies.

3.1. Deterministed Policy Gradient (DPG) random policy gradient

$$\nabla_\theta J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a\mathrm{d}s$$
$$= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]$$

This calculation shows that the gradient is an expectation of possible states and actions. Therefore, in order to obtain an approximate value of the gradient, a large number of samples need to be obtained from the action space and the state space.



actor-critic algorithm diagram

In order to better explore the environment, the DPG algorithm learns from the AC algorithm. Figure 1 shows the overall workflow of the AC algorithm. Specifically, the DPG algorithm is composed of actors and judges. The actors generate learning strategies, and the judges calculate the Q value function. In essence, the actor generates behavior in a given environment, and the judge generates a signal to judge the action. Then update commenters through TD learning, and update actors through policy gradients. Assume that the evaluation function parameter is ω, and the actor parameter is θ. The deterministic strategy gradient is:

ISSN: 2664-9640

$$\nabla_\theta J(\mu_\theta) = \int_S \rho^\mu(s) \nabla_\theta \mu_\theta(s) \left. \nabla_a Q^\mu(s,a) \right|_{a=\mu_\theta(s)} \mathrm{d}s$$
$$= \mathbb{E}_{s\sim\rho^\mu} \left[\nabla_\theta \mu_\theta(s) \left. \nabla_a Q^\mu(s,a) \right|_{a=\mu_\theta(s)} \right]$$

3.2. DQN network architecture

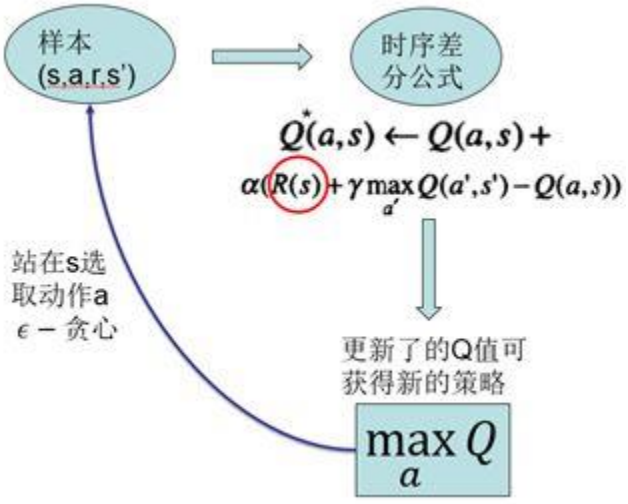
The DQN algorithm proposed by DeepMind in 2013 (an improved version of DQN was proposed in 2015) can be said to be the first successful combination of deep learning and reinforcement learning. If you want to integrate deep learning into reinforcement learning, there are some key problems that need to be solved, two of which are as follows:

1. Deep learning requires a large number of labeled data samples; while reinforcement learning is where the agent actively obtains samples, the sample size is sparse and delayed.

2. Deep learning requires that each sample is independent and identically distributed; and adjacent samples obtained by reinforcement learning are related to each other, not independent of each other.

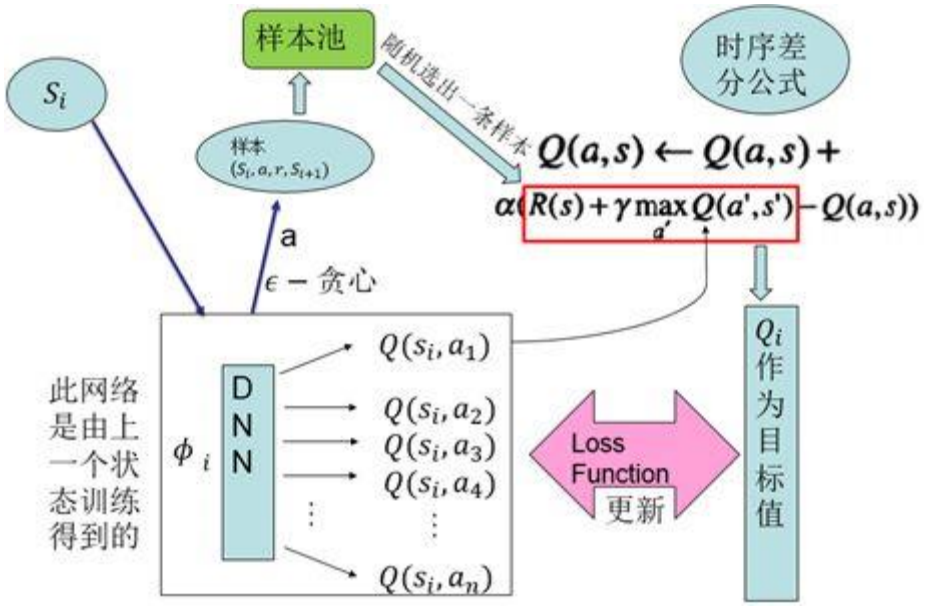
If you want to combine the two, you must solve the problems including the above two points.

Specifically, DQN is based on the classic reinforcement learning algorithm Q-Learning, a method of fitting the Q value with a deep neural network. The Q-Learning algorithm provides the target value of the deep network to update it. First look at the algorithm flow chart of Q-Learning:

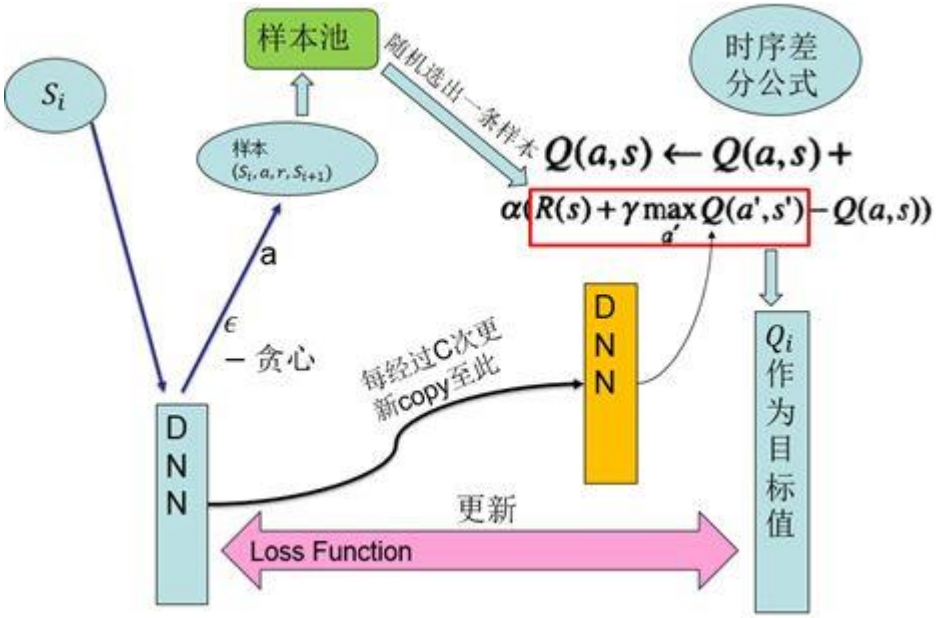


The agent adopts off-policy, that is, the implementation and improvement are not the same strategy, which is achieved through methods. Sampling in this way, and in the way of online update, the Q function is updated every time a sample is collected. The update is based on the timing difference formula. Get the new strategy with the updated Q function. The limitation of this classic reinforcement learning algorithm is that it cannot cope with high-dimensional input and cannot be applied to a large action space, especially, it cannot be applied to continuous action output. What DQN does is to use a deep neural network for end-to-end fitting, and give full play to the processing capabilities of the deep network for high-dimensional data input. The structure of the 2013 version is as follows:

ISSN: 2664-9640



Its improved version:



It has two key technologies:

1. Sample pool (experience reply): Put the collected samples into the sample pool first, and then randomly select a sample from the sample pool for network training. This treatment breaks the association between samples and makes them independent of each other.

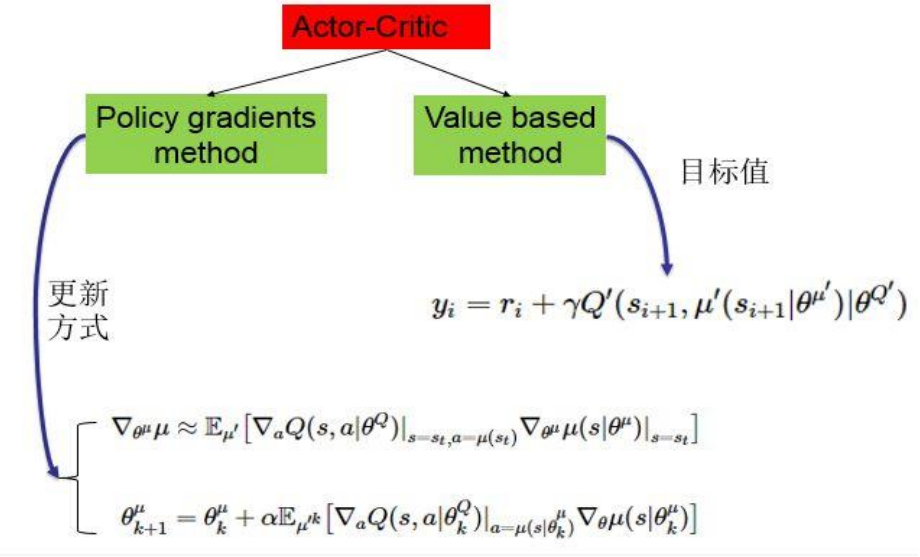
2. Fixed Q-target network: The existing Q-value is needed to calculate the network target value, and a network with slower update is used to provide this Q-value. This improves the stability and convergence of training.

3.3. Network architecture of DDPG

DQN is a method based on value function. The method based on value function is difficult to deal with the large action space, especially the continuous action situation. Because it is difficult for the network to have so many outputs, and it is difficult to search for the largest Q value among so many outputs. DDPG is based on the Actor-Critic method mentioned above. In terms of action output, a network is used to fit the strategy function, and the action is directly output, which can cope with the output of continuous actions and a large action space.

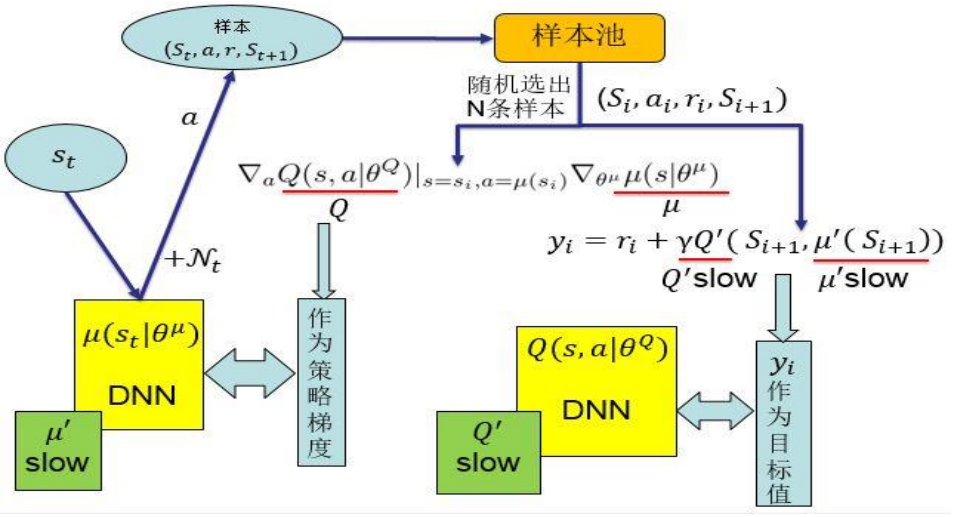
ISSN: 2664-9640

The structure of Acror-Critic is as follows:



The structure contains two networks, a policy network (Actor) and a value network (Critic). The strategy network outputs actions, and the value network judges actions. Both have their own update information. The strategy network is updated through the gradient calculation formula, and the value network is updated according to the target value.

DDPG uses the successful experience of DQN. That is, the two technologies of sample pool and fixed target value network are used. That is to say, these two networks each have a slower-changing copy, and the slower-changing network provides some values needed in the updated information. The overall structure of DDPG is as follows:



The DDPG method is another successful combination of deep learning and reinforcement learning, and is a very important research result in the development of deep reinforcement learning. It can deal with high-dimensional input, realize end-to-end control, and can output continuous actions, so that the deep reinforcement learning method can be applied to more complex situations with large action spaces and continuous action spaces.

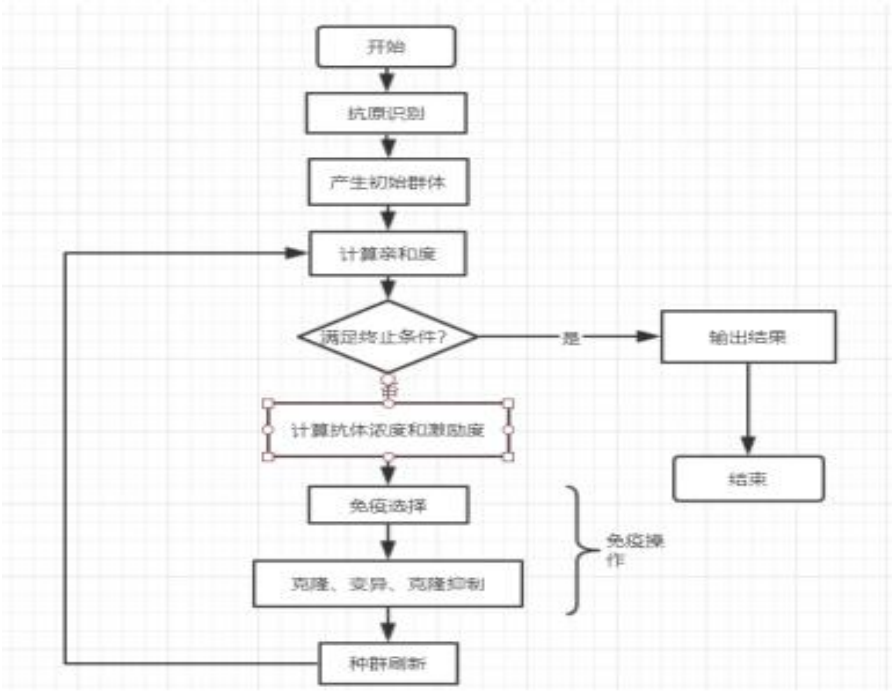
3.4. Introduction to immune algorithm

The immune algorithm is a new search-only algorithm inspired by the biological immune system. It is a heuristic random search algorithm that combines deterministic and random selection and has exploration and mining capabilities.

The main steps of the algorithm:

ISSN: 2664-9640

(1) Antigen recognition and initial antibody production;
(2) Antibody evaluation;
(3) Immune operation.
Features of immune algorithm:
(1) Global search capability;
(2) Diversity maintenance mechanism;
(3) Strong robustness;
(4) Parallel distributed search mechanism.
Immune algorithm flow chart:



Immune algorithm operator:
Affinity evaluation operator
For discrete coding, the best way is to use the Hamming distance of the antibody string:

$$aff(ab_i, ab_j) = \sum_{k=1}^{L} \partial_k$$

$$\partial_k = \begin{cases} 1, & ab_{i,k} = ab_{j,k} \\ 0, & ab_{i,k} = ab_{j,k} \end{cases}$$

Antibody concentration evaluation operator
The antibody concentration is usually defined as:

$$den(ab_i) = /frac1N/sum_{j=1}^{N} S(ab_i, ab_j)$$

In the formula, N is the population size, $S(ab_i, ab_j)$ represents the similarity between antibodies.
(3) Encouragement calculation operator
Antibody encouragement is the final evaluation result of antibody quality. Generally, antibodies with high affinity and low concentration will get greater encouragement.

ISSN: 2664-9640

$$sim(ab_i) = a \cdot aff(ab_i) - b \cdot den(ab_i)$$

$$sim(ab_i) = aff(ab_i) \cdot e^{-aden(ab_i)}$$

(4)Immune selection operator

The immune selection operator determines which antibodies enter the clone selection operation according to the excitation degree of the antibodies.

(5) Cloning operator

The clone operator replicates the selected individual antibody.

(6) Mutation operator

The mutation operator performs a mutation operation on the result of the clone operator to generate affinity mutations and realize local search. The mutation operator generates potential new antibodies in the immune algorithm, and is an important operator for realizing area search, which has a great influence on the algorithm. Different encoding methods require different mutation operators.

Discrete coding mutation operator: Based on binary, the value of bit is changed (inverted) to make it fall in the neighborhood of the source of discrete spatial variation.

(7) Clone suppression operator

The clone suppression operator is used to select clones after mutation to suppress antibodies with low affinity.

3.5. Gazebo function

1. Build a robot motion simulation model

In Gazebo, the three most basic objects, sphere, cylinder, and cube are provided. Using these three objects and their telescopic transformation or rotation transformation, you can design the simplest 3D simulation model of a robot. At the same time, Gazebo provides robot motion simulation. Through the plugin under Model Editor, we can add the algorithm files we need to verify, and then we can simulate the robot motion in Gazebo.

Build simulation models of various scenarios in the real world

Gazebo can create a simulation scene for testing robots. You can imitate the real world by adding object libraries, putting objects such as trash cans, ice cream bins, or even dolls, and you can also add 2D house plans through the Building Editor. Build a 3D house based on the design drawing.

Build a sensor simulation model

Gazebo has a very powerful sensor model library, including sensors commonly used by robots such as camera, depth camera, laser, imu, etc., and already has a simulation library, which can be used directly, or you can create a new sensor from 0 and add it. Specific parameters, even sensor noise model can be added to make the sensor more realistic.

Add physical properties of the real world to the robot model

There are force and physics options in Gazebo, you can add gravity, resistance, etc. to the robot. Gazebo has a physics simulation engine that is very close to the real one. Remember that the general ground has no resistance, which is different from the real world.

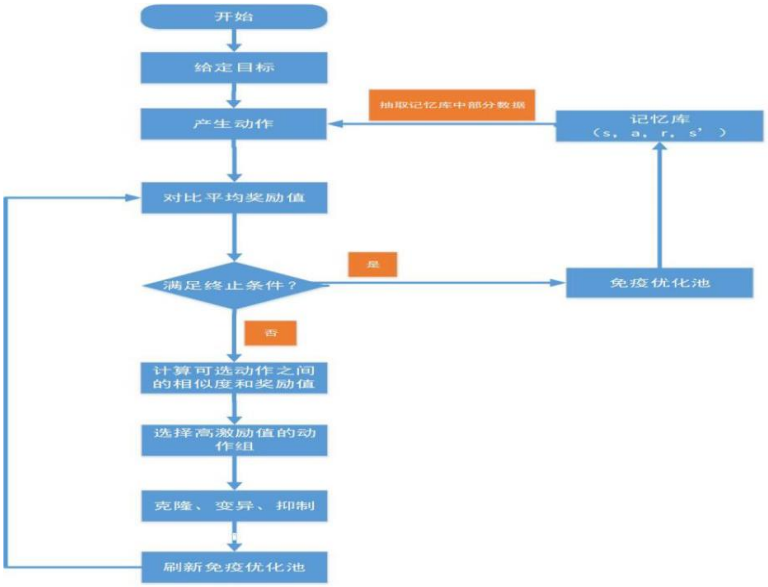
3 Based on improved ddpg autonomous navigation control algorithm

With the improvement of the learning ability of the DDPG algorithm, the fixed-size experience pool and sampling specifications cannot meet the algorithm's demand for multi-feature data, which limits the learning rate of the algorithm. In order to solve the above problems, this paper proposes the DDPG-vcep (Variable capacity experience pool) algorithm. By adding immune learning to the DDPG algorithm, the algorithm realizes real-time adjustment of the experience

ISSN: 2664-9640

pool capacity and sampling specifications according to its own learning curve, and improves the algorithm's sample size. Data utilization.

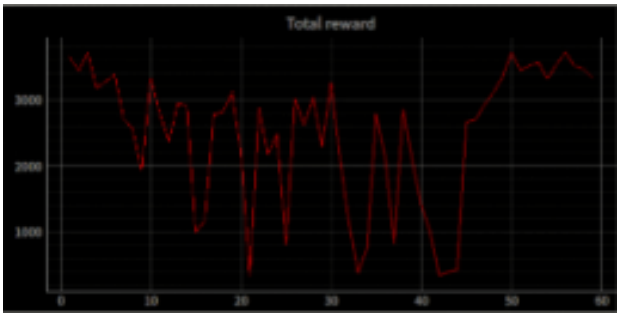
The flow chart of DDPG combined with immune learning algorithm is as follows:



4. Experimental results

4.1. Experimental setup

Choose Ros/Gazebo as the simulation environment. Other software include Anaconda2.7, Keras and Tensorflow v1.14. The experimental hardware is Ubuntu 16.04 system, GPU is GTX-1070Ti, 12-core CPU and 32G running memory. The model of turbot3 is imported into gazebo. The white dots are the designated targets. Arrive will be output when the designated target is reached. First, classify high rewards and low rewards in the ddpg experience memory pool according to the average reward value, and then try to copy and save the memory group with high reward value, perform the immune operation on the saved memory group, and compare the immune operation memory group with The memory group with high reward value is combined to obtain the optimized memory group to achieve rapid training of the strategy model library. Here is the drawing tool used before running dqn:

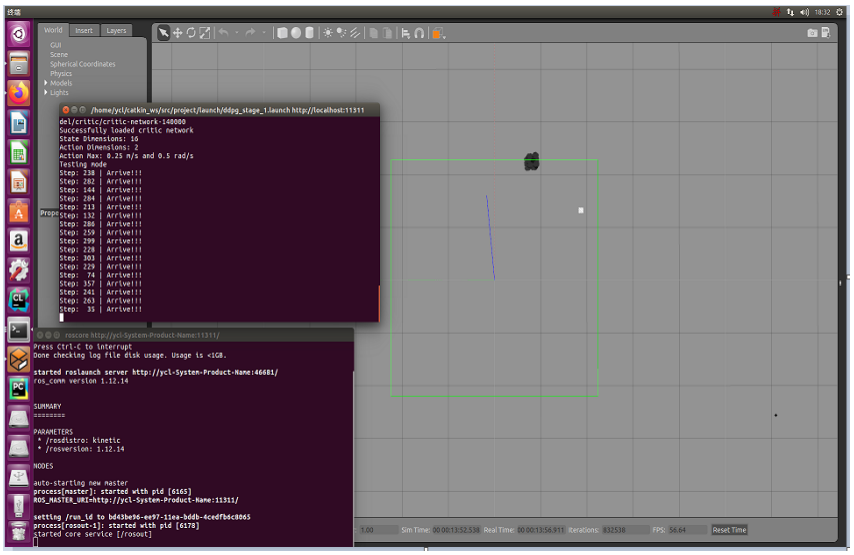


Changes in the total reward curve before the improvement

4.2. Conclusion analysis

The following figure shows the effect of the strategy model after the algorithm is improved. The probability of reaching the target point is 100% and the curve change of the total reward:

ISSN: 2664-9640



The probability of reaching the target point is 100%

Changes in the reward curve after the improvement

Deep reinforcement learning is a process of training models and using training models. We previously needed 140,000 batches to train the available models to achieve 100% accuracy. However, the process of model training convergence can be accelerated by improving the memory library of the algorithm. It takes 120,000 batches to achieve a 100% correct rate.

5. Conclusion

In order to make the data of RL (data whose relevance is not independently distributed) closer to the data required by DL (data with independent and identical distribution), a "memory" is established during the learning process, and the state, action, and state_ (The next moment state) and reward are stored in the memory bank. Each time the neural network is trained, a batch of memory data is randomly selected from the memory bank, which disrupts the order of the original data and weakens the relevance of the data. This paper chooses to optimize the data in the memory bank through the rules of the immune learning algorithm so that the best model can be obtained quickly.

References

[1] Tai L,Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation[C]//Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017: 31-36.

ISSN: 2664-9640

[2] Tail L, Zhang J, Liu M, et al. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 1111-1117.
[3] Lei T, Ming L. A robot exploration strategy based on q-learning network[C]//Real-time Computing and Robotics (RCAR), IEEE International Conference on. IEEE, 2016: 57-62.
[4] Zhu Y, Mottaghi R, Kolve E, et al. Target-driven visual navigation in indoor scenes using deep reinforcement learning[C]//Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017: 3357-3364.
[5] Mirowski P, Pascanu R, Viola F, et al. Learning to navigate in complex environments[J]. ar Xiv preprint ar Xiv:1611.03673, 2016.
[6] Mirowski P, Grimes M K, Malinowski M, et al. Learning to Navigate in Cities Without a Map[J]. ar Xiv preprint ar Xiv:1804.00168, 2018.