2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT) | 978-1-6654-5087-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/AICIT55386.2022.9930247

Path planning of improved DQN based on quantile regression

Lun Zhou
School of Electrical and information Engineering
Wuhan Institute of Technology
Wuhan, China

Ke Wang
School of Electrical and information Engineering
Wuhan Institute of Technology
Wuhan, China

Hang Yu
School of Electrical and information Engineering
Wuhan Institute of Technology
Wuhan, China

Zhen Wang*
School of Electrical and information Engineering
Wuhan Institute of Technology
Wuhan, China
wangzhen@wit.edu.cn

***Abstract*—To solve the problems of slow convergence and overestimation of the value of quantile regression-deep reinforcement learning algorithm, a Dueling Double Depth Q algorithm based on quantile regression (QR-D3QN) was proposed. Based on QR-DQN, the calculation method of the target Q value is modified to reduce the influence of value overestimation. Combining the confrontation network and adding preferential experience sampling to improve the utilization efficiency of effective data. It is verified by the ROS-Gazebo simulation platform that the robot can effectively select actions, get a good strategy, and can quickly avoid obstacles and find the target point. Compared with D3QN, the route planned by the robot is shortened by 4.95%, and the obstacle avoidance path is reduced by 18.8%.**

Keywords—Path planning, quantile regression, prioritized experience replay, deep reinforcement learning, dueling network

I. Introduction

The algorithms for robot path planning in unfamiliar environment are usually traditional mathematical methods, such as A* algorithm, ant colony algorithm and related derivative algorithms[1]. Tai Lei applied the deep reinforcement learning algorithm to this field for the first time [2].

The purpose of reinforcement is to make an object acquire the ability to complete a certain task independently. Sarsa[3] and Q-Learning[4] are the basic algorithms of deep reinforcement learning. DQN uses neural network instead of "experience base" in reinforcement learning, which solves the problem of data explosion in complex environment[5]. Subsequent DDQN and Dueling-DQN algorithms are constantly improving DQN. These algorithms generally select actions based on the expectation of value, ignoring the distribution of value. DeepMind researchers proposed C51 based on the distribution of value[6]. C51 changes the output of neural network from the expectation of action value to the probability distribution of action value. The probability that the output of the neural network corresponds to 51 values. C51 proved that the bellman equation based on probability accords with Markov decision process[6], and the Wasserstein metric between probability distributions can converge. However, because the Wasserstein metric can't be directly optimized by the method of random gradient descent, C51 didn't realize this theory in , but used KL divergence to measure the difference of distribution and minimize it.

Based on C51, the QR-DQN is produced. The transposes C51, changing the original divided value into evenly distributed quantiles, and the output of the neural network is the action value. QR-DQN changes from fixed value and adjusted probability to fixed probability and adjusted value. Furthermore, quantile regression is proposed to describe the Wasserstein metric, so that gradient descent can be used to optimize the Wasserstein metric[7].

As the choice of action is still based on the value of action, QR-DQN also has similar problems to DQN. On the basis of QR-DQN and some improved methods of DQN , this paper proposes QR-D3QN .

II. Related Work

A. Distributional RL

Distributed reinforcement learning also conforms to the Markov decision process (s, a, R, P, γ)[8]。 S is the state space, a is the action space, R is the immediate reward of action, and P (s' | s, a) is the probability of taking action a to change from state s to state s', γ Is the discount coefficient of the reward. The task of completing a round is to make a series of actions to form a strategy π. The general din measures the action value function of the strategy as[9]:

$$Q^{\pi}(s,a) = E[R(s,a)] + \gamma E_{\pi}[Q(s',a')]] \tag{1}$$

The current value is equal to the immediate reward of the current action plus the discount value of the next action. Using the expectation of value to calculate itself ignores a lot of information. QR-DQN introduces quantiles and replaces the expectation of value with the distribution of value, covering more internal random values. It should be emphasized that in the definite state space and limited action space, the internal value is fixed rather than uncertain, but random. This value can be similarly described using the distributed Behrman equation[7]:

$$T^{\pi} Z(s,a) = R(s,a) + \gamma Z(s',a') \tag{2}$$

$T^{\pi}Z$ represents the value mapped to the quantile. Other calculations are still based on the time difference method. C51 is based on distributed bellman equation and has better performance in Atari 2600[6].

B. The Wasserstein Metric

The Wasserstein metric is a concept to measure the gap between two probability distributions[10]. Its narrow definition is:

$$\mathrm{W}(\mathrm{U},\mathrm{Y}) = \min \textstyle\sum_{\mathrm{x_u},\mathrm{x_y}} \gamma(\mathrm{x_u},\mathrm{x_y}) \|\mathrm{x_u} - \mathrm{x_y}\| \tag{3}$$

The sum of the probabilities is 1, and there is more than one scheme to change from distribution U to distribution Y. As shown in Figure 1, the probability of one place in distribution U can be transferred to any place in distribution Y, and the scheme with the least amount of movement is called the Wasserstein metric.

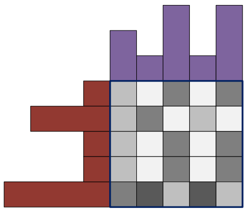

Fig. 1. Distribution of The Wasserstein Metric, the darker the gray, the greater the gap.

Based on the Wasserstein metric, C51 evenly divides the minimum and maximum values of value into 51 equal parts, which are mapped to the probability from 0 to 1. Optimization is the process of probability adjustment, which minimizes the gap between the probability distribution of the current value and the probability distribution of the target value. However, when the Wasserstein metric is directly decreased by a random gradient, gradient $E_I d_p(P_i^u, P_i^y)$ obtained by random gradient descent is not equal to the original gradient[6]:

$$\begin{aligned} d_p(U,Y) &= d_p(\sum_i A_i P_i^u, \sum_i A_i P_i^y,) \\ &\le \sum_i d_p(A_i P_i^u, A_i P_i^y) \\ &\le \sum_i Pr\{I=i\} d_p(P_i^u, P_i^y) \\ &= E_I d_p(P_i^u, P_i^y) \end{aligned} \tag{4}$$

A is the indicator scalar mapped to probability. Therefore C51 uses KL divergence to measure the distribution gap.

KL divergence originates from the entropy of thermodynamics, which excludes the redundant information of data and can only measure the gap between the corresponding positions of distribution. The Wasserstein metric is a real probability measure, which calculates the probability gap of each position in the distribution so that the potential information can be retained.

QR-DQN aims to solve the problem that the distance of bulldozers can't be decreased by the random gradient. The value of the action is projected to the quantile, and the quantile is evenly divided into i equal parts from 0 to 1. Every quantile τ_i corresponds to a z_i, which means $P(z_i<z)=\tau_i$, as shown in Figure 2. When the network parameters are iterated, QR-DQN adjusts the value and fixes the quantile. At this time, the quantile does not directly participate in the gradient decline, thus avoiding the gradient deviation.

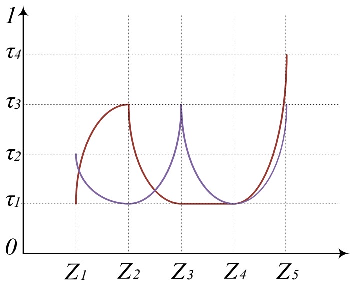

Fig. 2. Reward distribution.

C. Improved Method of DQN

To solve the problems of over-estimation and local optimal solution of the DQN, Double DQN[11] and Dueling DQN[12] appeared one after another. The former mainly modifies the calculation method of the target value, from selecting the maximum value action according to the target network to selecting the maximum value action according to the current network:

$$Q_t = R + \gamma Q_t'(s', argmax_a Q'(s', a'; \theta); \theta_t) \tag{5}$$

This effectively weakens the correlation of data. The latter puts forward that in many deep reinforcement learning tasks based on vision, the cost function is only related to the state and has nothing to do with the action. The network constructed by this theory is called a confrontation network [12]. The network connects the output of the convolution layer to two fully connected layers. One layer fits the state value, and the other layer fits the additional action value:

$$\mathrm{Q}(s,a;\alpha,\beta,\theta) = \mathrm{V}(s;\beta,\theta) + (\mathrm{A}(s,a;\alpha,\theta) - \frac{1}{|N|}\sum \mathrm{A}(s,a;\alpha,\theta)) \tag{6}$$

α represents the parameter of the action-full connection layer; β represents the state value-the parameter of the full connection layer; θ represents a common parameter; N is the number of actions.

III. Path Planning of QR-D3QN

The overall process of QR-D3QN path planning is shown in Figure 3. Get the image and radar information from the environment as the current state S. Then, according to the ε -greedy algorithm, enter the exploration or utilization process, and select the current action A. If it is the utilization process, input S into the neural network to get the action value Q, otherwise, take random action. Then get the instant reward R and the following state s', and store (s, a, r, s') in the experience pool. Then, a batch of batch-size (s, a, r, s') is extracted from the experience pool to train and update the neural network parameters through priority experience playback. Finally, s' is taken as the current state, and so on until the training is finished.

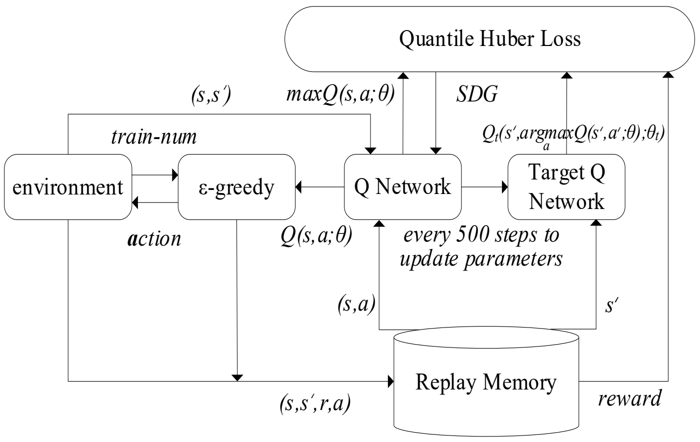

Fig. 3. The frame diagram of QR-D3QN algorithm.

A. Environmental Information Fusion

In the Gazebo simulation environment, the position and velocity information of each object can be obtained simply by command. The distance between the carrier and the target point and the angle between the carrier speed and the target

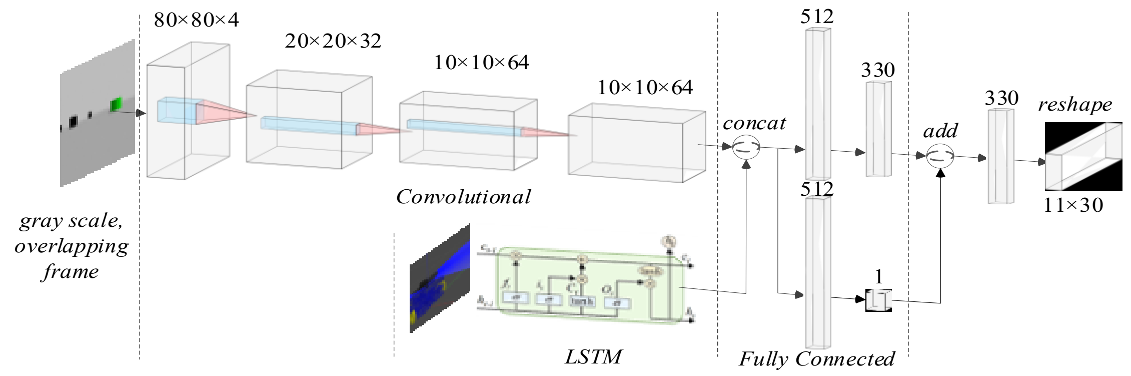

Fig. 4. Structure block diagram of QR-D3QN environmental information fusion network

point constitute the state information of the carrier, which affects the movement of the carrier to the target point in each action cycle.

The original camera image belongs to the ROS Image message type, with a resolution of 1024×728. Before being input to CNN, it is superimposed and optimized four times and becomes a gray image of 80×80×4. The image can distinguish the target point from the obstacle, but can not directly distinguish the distance between the carrier and the target point and the obstacle, so lidar is added. The laser radar belongs to the message type of ROS LaserScan, and its effective range is [-90, 90] at 0 o'clock in front of the carrier, and its effective distance is [0.1m,10m]. The generated data belongs to sequence information and is processed by LSTM (Long Short Term Memory Network).

B. *Network Structure*

The image is spliced with the radar information processed by LSTM through a three-layer convolution operation, and then based on Dueling DQN's countermeasure network, two completely connected layers with the same dimension are connected. One represents the first state value layer, the other represents the first additional action value layer and then connects the two full connection layers respectively. The shape of the two outputs is [batch-size, 1] and [batch-size, action-num×quantity-num] respectively. At last, calculate the action value [batch-size, action-num, quantify-num].

C. *Loss Function Based on The Wasserstein Metric*

Calculate general quantile τ:

$$\min_{\xi} \begin{matrix} \sum_{y_i \geq \xi(x_i,\beta_\tau)} \tau(y_i - \xi(x_i,\beta_\tau)) + \\ \sum_{y_i < \xi(x_i,\beta_\tau)} (1-\tau)(\xi(x_i,\beta_\tau) - y_i) \end{matrix} \tag{7}$$

β is the equation coefficient.

The output of the neural network is the result of quantile regression. τ takes a uniform value from 0 to 1, indicating the distribution of value. The outputs of the target network and the current network are two-dimensional arrays, the shape is [action-num, quantify-num]. To find the distribution gap between them, it is necessary to traverse and subtract each value.

QR-D3QN adopts Huber Loss[13]:

$$L_k = \begin{cases} \frac{1}{2}u^2, & if|u| \leq k. \\ k(|u| - \frac{1}{2}k), & otherwise. \end{cases} \tag{8}$$

u is the difference in network output distribution. The loss function is smoother in the [-k, k] interval. The value obtained by the Huber loss is multiplied by the formula (7) τ or (1- τ)。

Based on the priority experience playback[14], when the batch-size of the training is not 1, the final L needs to be multiplied by the weight corresponding to the sample:

$$\omega = (\text{batch\_size} \times \frac{\text{TD\_Loss}}{\sum_N \text{TD\_Loss}})^{-(0.4+\frac{0.6}{N})} \tag{9}$$

N is the number of training steps.

This makes the probability of the samples with smaller Loss participating in the update higher, thus improving the utilization rate of effective information.

IV. SIMULATION RESULTS AND ANALYSIS

A. *Action Partition*

The car is used as the carrier in the simulation, and the motion control is the control of linear velocity and angular velocity rather than the direction control of fixed speed. Using steering instead of translation has a better reference for practical design. The speed is divided into 11.

TABLE I. ACTION PARTITION TABLE

action number	**linear velocity/ ($m \cdot s^{-1}$)**	**angular velocity/ ($rad \cdot s^{-1}$)**
1	1	-1
2	1	-0.5
3	1	0
4	1	0.5
5	1	1
6	0.5	-1
7	0.5	0
8	0.5	1
9	0	-1
10	0	0
11	0	1

B. *Rewards and Hyperparameters*

TABLE II. SETTING OF REWARD FUNCTION

state	**reward**
Enter within 1.5m of obstacles	-1
Away from the target	$-0.1 \times (d_t - d_{t-1})$
Close to the target	$0.1 \times (d_{t-1} - d_t)$
Linear or angular velocity changes	$-0.01 \times (\omega_t - \omega_{t-1} + v_t - v_{t-1})$
Enter within 1m of the target	20

The immediate rewards of actions are shown in Table II. The algorithm always records the current speed and the previous time speed and the distance between the carrier and the target point at the current time and the previous time.

The algorithm and environment-related parameters are shown in Table III.

TABLE III. INITIAL PARAMETER TABLE

parameter	value
γ	0.99
Learning_rate	0.0001
Greedy policy termination parameter: $\varepsilon_{\text{final}}$	0.1
Training times	435000
Experience pool size	40000
Number of quantiles	30
Target Network update interval step size	800
batch size	32

C. *Experiments and Results*

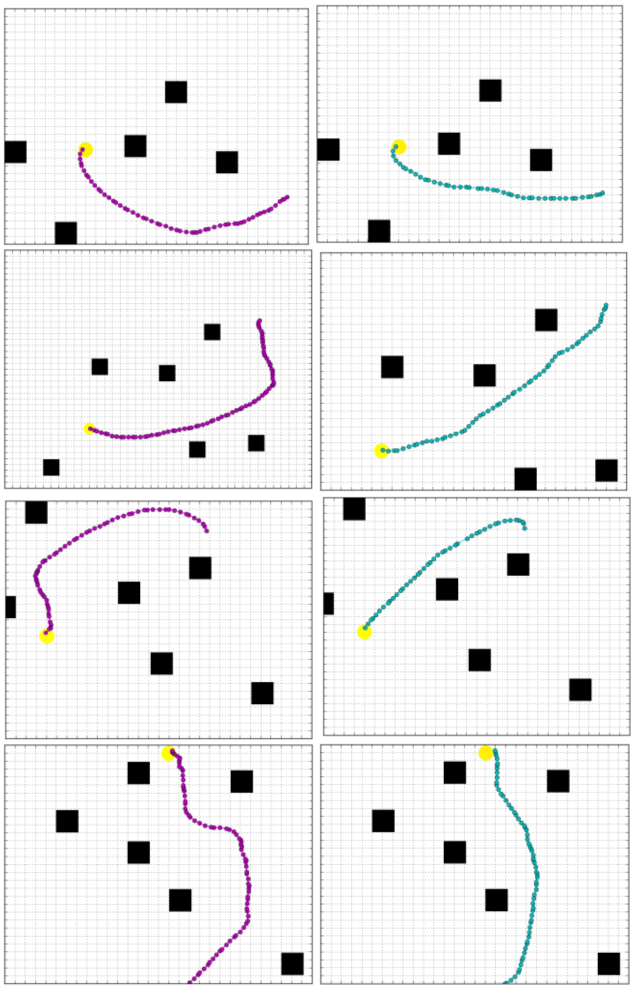
Fig. 5. Path comparison diagram, purple on the left is D3QN, and cyan on the right is QR-D3QN. The black square is an obstacle, and the yellow circle represents the target point.

The experiment adopts ros1-gagebo7 joint simulation based on GTX 1660, and the map size is 30 meters× 30 meters. At the beginning of each simulation, the starting position, obstacle position and target point of the car are generated. The target point is a cylinder with a radius of 0.5 meters. The obstacle is a cube with a length of 1m and an influence radius of 1.5 meters. The distance between the starting position of the trolley and the target point is not less than 20 meters.

On the car, there are cameras and lidar, which are used as the inputs of the neural network. The D3QN algorithm and QR-D3QN algorithm are used to test and compare.

There are cameras and lidar on the car, which are used as the inputs of the neural network. D3QN algorithm and QR-D3QN algorithm are used to test and compare.

In a total of 6,000 simulation steps, D3QN can only complete 99 rounds, and QR-DQN can complete 113 rounds. On average, the route length of QR-DQN saves 4.95%. The average obstacle avoidance distance can be saved by 18.8% if the linear distance between the starting point and the target point is subtracted and only the obstacle avoidance distance is considered. The route planning diagram of QN and QR-D3QN is shown in Figure 5.

REFERENCES

[1] LIU Zhi-rong, JIANG Shu-hai, Review of mobile robot path planning based on reinforcement learning, Manufacturing Automation, 2019, vol.41, pp.90-92.

[2] Tai Lei, Liu Ming, “A Robot Exploration Strategy Based on Q-learning Network”, IEEE RCAR. Cambodia, 2016.

[3] Sutton, R. S., Generalization in reinforcement learning: Successful examples using sparse coarse coding, In Advances in neural information processing systems, 1996, pp. 1038-1044.

[4] CJCH Watkins, Learning From Delayed Rewards, King's College Londo, 1989.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, “Playing Atari with Deep Reinforcement Learning,”, arXiv:1312.5602, 2013.

[6] Marc G. Bellemare, Will Dabney, Rémi Munos, “A Distributional Perspective on Reinforcement Learning”, arXiv:1707.06887, 2017.

[7] Will Dabney, Mark Rowland, Marc G. Bellemare, Rémi Munos, “Distributional Reinforcement Learning with Quantile Regression”, arXiv:1710.10044, 2017.

[8] Bellemare, M. G., Dabney, W., and Munos, R, “A Distributional Perspective on Reinforcement Learning”, Proceedings of the 34th International Conference on Machine Learning (ICML), 2017.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Human-level control through deep reinforcement learning, NATURE, 2015, vol.518, pp.529-542.

[10] Bellemare, M. G, Danihelka, I, Dabney, W, Mohamed, “The Cramer Distance as a Solution to Biased Wasserstein Gradients”, arXiv.

[11] Hasselt H, Guez A, Silver D, “Deep reinforcement learning with double q- Learning”, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, USA, 2016, pp.2094-2100.

[12] Ziyu Wang, Tom Schaul, Matteo Hessel, “Dueling Network Architectures for Deep Reinforcement Learning”, arXiv:1511.06581 [cs.LG], 2015.

[13] Aravkin, A. Y, Kambadur, A, Lozano, A. C, Luss, R, “Sparse Quantile Huber Regression for Efficient and Robust Estimation”, arXiv.

[14] Schaul T, Quan J, Antonoglou I, et al, “Prioritized experience replay”, arXiv: 1511.05952, 2015.