ISSN: 2664-9640

Application of D3QN algorithm based on behavior exploration strategy

Yuhua Li, Jian Li

School of Sichuan University, Chengdu 610000, China

Abstract

Although reinforcement learning has developed rapidly and its applicability has been greatly improved, the reinforcement learning algorithm attaches great importance to the adaptability of the environment.The performance of the same algorithm varies in different environments, and fine-tuning of hyperparameters may also affect the experimental results. Aiming at the problem of insufficient stability of D3QN algorithm, combined with the behavior exploration strategy of TD3 algorithm, this paper proposes a pseudo-gaussian action exploration scheme based on discrete action space and summarizes four algorithm training skills.Then, the improved algorithm was compared with the classical algorithm D3QN and TD3 in different scenes of gym, and good results were obtained, which verified the feasibility of the improved algorithm.

Keywords

Reinforcement learning, explorethepossibilities, advantage function.

1. Introduction

Reinforcement Learning (RL) is a special machine Learning method in which an agent interacts and trial-and-error with the environment, and evaluates the mapping relationship between states and actions according to the received returns [1, 2]. Reinforcement learning has the advantages of self-learning and online learning and is one of the core technologies for designing intelligent autonomous systems. At present, with the continuous improvement and development of theory, reinforcement learning technology is more and more applied to industrial control, job scheduling, production management and other aspects, and gradually become a research hotspot in the field of machine learning.

This paper is based on Dueling Double DQN (D3QN) algorithm to improve the agent behavior exploration strategy. We know D3QN applied to discrete environment, is based on the Q value to study the typical algorithm, so the complexity of the algorithm is low, run faster. However, with the increase of action dimension, its ability to learn the environment decreases, and it may even fail to learn effective actions in a complex environment, resulting in non-convergence of results; TD3 is usually used in scenes of continuous action [3]; It is an excellent algorithm based on actor-critic architecture, which can be competent for most scenes without any adjustment and has strong portability. However, as this algorithm is a deterministic strategy algorithm, once the rewards are sparse, there will be a situation of insufficient exploration, unable to jump out of the local optimal strategy, and ultimately unable to complete the environmental objectives. In this paper, based on D3QN framework and TD3's behavior exploration strategy, a variant algorithm of D3QN (SDQN) is proposed by using four-point algorithm training skills.

2. Method

2.1. Overview of D3QN algorithm

In 2016, Google DeepMind proposed Dueling Network Architectures for Deep Reinforcement Learning, which uses advantage function, After collecting only one discrete action data, Dueling

ISSN: 2664-9640

DQN can more accurately estimate Q value and select more appropriate actions [4]. Double DQN, on the other hand, determines the choice of action by introducing another value function with the same structure as the target network, thus eliminating the problem of overestimation of Q value [5]. Obviously, these two algorithms are based on DQN, and it is natural to think of the combination of the two algorithms, and thus get the D3QN algorithm, which has the advantages of Dueling DQN and Double DQN. The algorithm is briefly described as follows:

1) In traditional DQN learning, overestimation exists in the prediction of Q(s', a').The simple idea is to use a value function network to calculate all the action evaluation value Q(s', a') of the next state and get the maximum value, and then use the time difference idea to get the target Q value of the current state.The expression is as follows:

$$Q_t^{DQN} = R_{t+1} + \gamma \max Q(S_{t+1}, a; \theta_t^-) \tag{1}$$

It can be seen that the maximization operation is used to obtain the Q value, while the neural network has an error in predicting Qmax, which makes the neural network overestimate the value function in the constant iteration of parameter update, resulting in overestimate. Therefore, the idea of Double DQN is to use another neural network with the same structure to reduce the maximum value of the approximation function and reduce the error. Although it still adopts the maximization operation when selecting target Q value estimation, it separates the selection action from the evaluation action. The final expression is as follows:

$$Q_t^{DDQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max Q(S_{t+1}, a; \theta_t); \theta_t^{'}) \tag{2}$$

2) In order to obtain the value estimation function of the environment state better, the DQN neural network is modified to some extent to improve the expression ability of the value function. The main methods are as follows: Decompose the state-action value function Q(s, a) to obtain the state value function V(s) and the action advantage function A(s, a), and construct and allocate the corresponding neural network for them respectively. The network output combination of these two represents Q(s, a). The advantage function of the action can be expressed as:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{3}$$

The state value function V(s) reflects the estimated value of the current state of the agent, while the state-action value function Q(s, a) represents the estimated value of the environmental return that the agent can obtain by choosing an action in a certain state. As for the advantage function, it describes the estimated size of different actions taken by an agent in a certain state, reflecting the quality of each action.Therefore, an expression of the following form can be obtained:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)) \tag{4}$$

2.2. Improvement of D3QN algorithm

In order to improve the efficiency of discrete action space exploration, a discrete action of approximating gaussian exploration scheme is proposed based on the behavior exploration strategy of TD3 algorithm. The Form of gaussian function is:

$$f(x) = ae^{-(x-b)^2/2c^2} \tag{5}$$

where a, b and c are constants, and a> 0.

As can be seen from the properties of Gaussian function, if it is used for sampling, the sampled data are mostly concentrated around the mean value, and the trend becomes more obvious with the decrease of variance. For an agent corresponding to reinforcement learning, its behavioral exploration strategy needs to explore as many environmental states as possible while considering the utilization of experience samples when interacting with the environment. And

ISSN: 2664-9640

that's exactly what the Gaussian does. We know that in the environment of continuous action, we can directly add a Gaussian noise to the output of actor neural network to achieve. However, in discrete action environment, because the output of policy network is usually a natural array, which represents the i+1 element of discrete action space should be selected, it is difficult to directly add gaussian noise to achieve. However, different probability maps of [-2, -1, 0, 1, 2] can be obtained by using random uniform sampling function and simple logic rules. Finally, a truncation operation can be performed after superimposing with the output of the policy network to achieve the effect similar to gaussian function. Its expression is as follows:

$$f(x)=\begin{cases}0 & ,x\geq a\\ 1 & ,a>x\geq b\\ -1 & ,b>x\geq c\\ 2 & ,c>x\geq d\\ -2 & ,else\end{cases} \tag{6}$$

where x is the uniform random sampling value in the interval of (-1, 1). And a, b, c, d must meet the following conditions: The probability of obtaining 0 is the highest, ±1 is the second, and ±2 is the lowest. The final approximate gauss strategy exploration action value is:

$$A=(f(x)+\arg\max_{a}(Q(s,a;\theta)).clip(0,n-1) \tag{7}$$

Where n is the length of discrete action space.Compared with the Epsilon-greedy strategy, this approximate Gaussian discrete space exploration method has the advantage that it can make better use of the samples explored by agents and concentrate on learning the behavior value function near the greedy strategy as soon as possible, so as to improve the convergence speed of the algorithm.In addition, in order to further improve the performance of the algorithm, four techniques are summarized as follows:

1) Before the agent exploration, the uniform random exploration strategy with a certain number of steps should be completed.

2) The learning mode of the algorithm is changed from per-step update to post-turn update, and the times of each update are the corresponding times of interaction with the environment at the current round.

3) Discretization of state or action space without changing the environment model.

4) Change the update mode of target network to soft update by referring to TD3 algorithm.

3. Experimental simulation and result analysis

In order to verify the effectiveness of the algorithm, the improved algorithm is compared with D3QN and TD3, and a series of algorithm simulation experiments are conducted (the environment adopts Classic control and Box2D scenes of gym). The specific application scenarios are CartPole, MountainCar, Pendulum and LunarLander.

In order to make the experimental results more comparative and convincing, the parameters used by each algorithm should be the same as much as possible, so as to minimize or avoid fine-tuning effects of hyperparameters.In addition, the learning factor lr=0.0005, return attenuation factor γ=0.99, training sampling packet size batch_size =100, network hidden layer hid =128, network update attenuation factor τ=0.005 in all experimental algorithm parameters.The neural networks constructed are all three-layer fully connected structure, and the activation function of all the other layers except the output layer is relu function (the output layer activation function of TD3 algorithm is tanh function). The unique parameters of the other algorithms are adjusted to make the algorithm perform as well as possible.In addition, 10 experiments were conducted for each scene, and the return learning curve of each experiment was added and averaged, and a smoothing process with a window size of 4 was performed as the return learning curve of the final algorithm. In this way, the accidental performance of the

ISSN: 2664-9640

algorithm is avoided or reduced, and more attention is paid to the stability of the algorithm. Note: If the curve is the final target return value and is a straight line, it indicates that the algorithm has been tested for 10 times in a certain environment and the success rate of completing the task is 100%. The greater the difference of return between goal and the final completion of the task, the lower the success rate of the algorithm and the worse the effect.

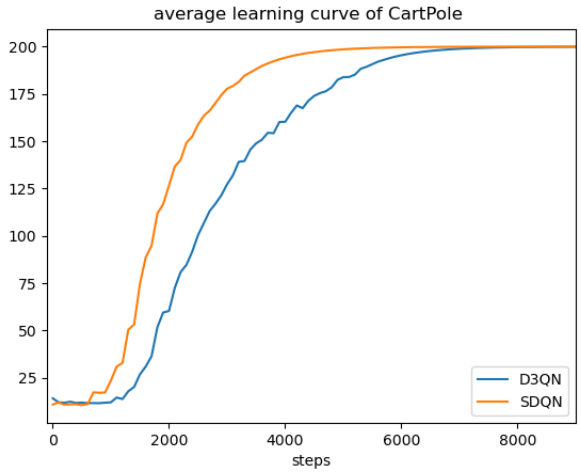

Fig 1: CartPole discrete scene

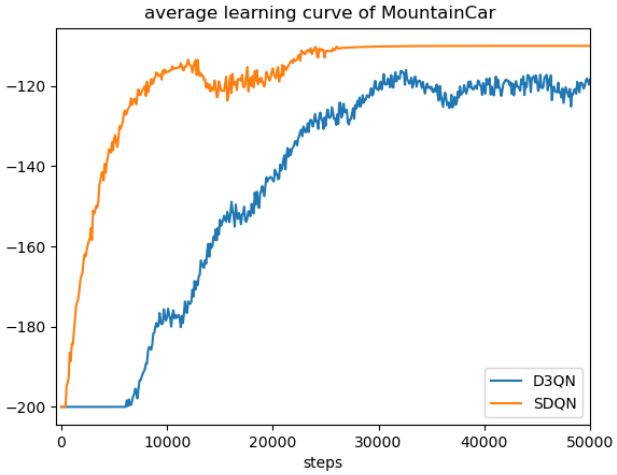

Fig 2: MountainCar discrete scene

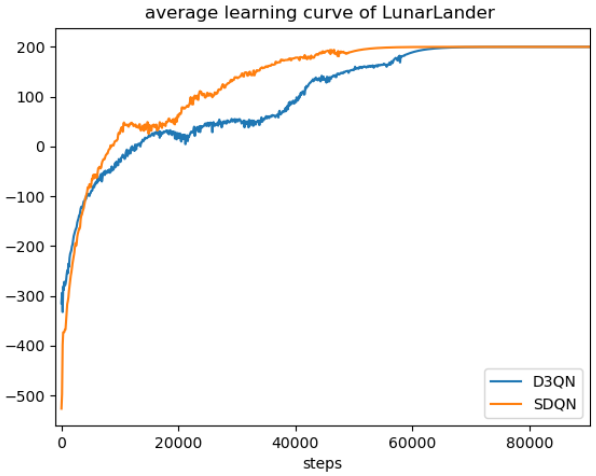

Fig 2: LunarLander discrete scene

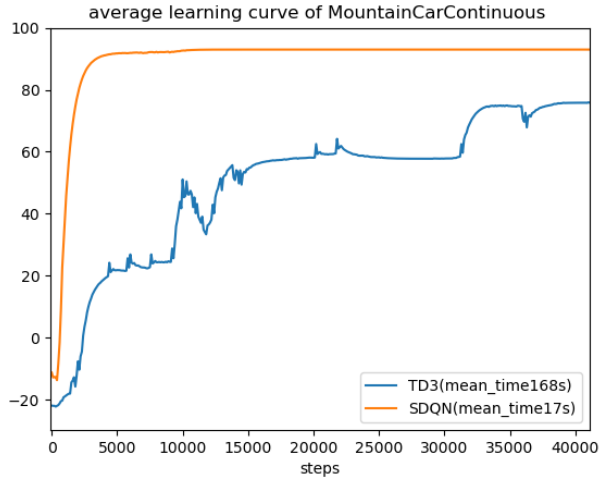

Fig 3: MountainCar continuous scene

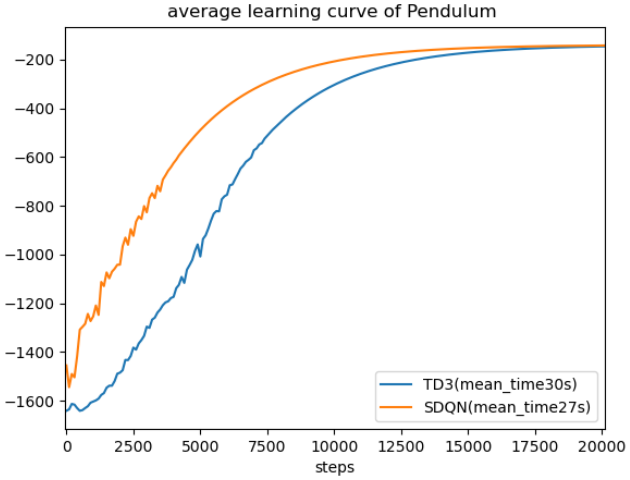

Fig 5: Pendulum continuous scene

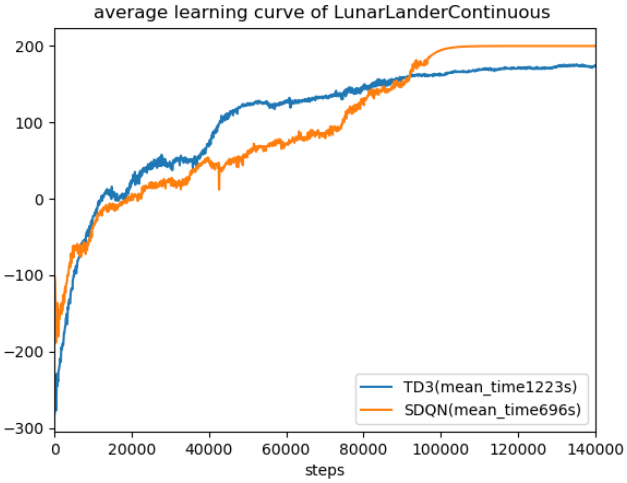

Fig 6: LunarLander continuous scene

ISSN: 2664-9640

As can be seen from the comparison diagram of average cumulative returns of the six algorithms above, SDQN can complete environmental tasks better than TD3 and D3QN algorithms in different environments. The success rate of the algorithm is 100%, and the convergence speed is faster than the other two algorithms.D3QN's average cumulative reward in MountainCar's action-discrete environment was lower than the task target, indicating that the success rate was less than 100%. Meanwhile, the task completion rate of TD3 is lower than that of SDQN in MountainCar and LunarLander's action continuous environment. In particular, due to the issue of sparse rewards in the MountainCar environment, rewards of the scene was reshaped.

In short, SDQN combines the advantages of both. It can not only deal with discrete and continuous action environments, but also accelerate the effective utilization rate of action exploration, and further promote the convergence speed and stability of the algorithm.However, it also has the characteristics of poor exploration and cannot deal with higher dimensional continuous action environment, that is, it has poor processing for more complex high-dimensional environment or sparse reward scenes [6].

4. Conclusion

To solve the problem that the reinforcement learning algorithm needs to adjust the hyperparameters frequently and the convergence effect is relatively unstable when it is applied to different simulation environments, the behavior exploration strategy of D3QN algorithm applied to discrete action space is optimized by the idea of gaussian function and logical rules. In addition, the stability of the algorithm is further enhanced by four techniques that are easy to implement and hardly increase the complexity of the algorithm, so that the new algorithm can reach 100% success rate in different environments.

References

[1] T. Yang, H. Tang, C. Bai: Exploration in Deep Reinforcement Learning: A Comprehensive Survey. arXiv preprint arXiv:2109.06668, 2021.

[2] R. S. Sutton, A. G. Barto: *Reinforcement learning: An introduction*. MIT press, 2018.

[3] S. Fujimoto, H. van Hoof, D. Meger: Addressing function approximation error in actor-critic methods. arXiv preprint arXiv:1802.09477, 2018.

[4] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas.Dueling Network Architectures for Deep Reinforcement Learning. arXiv preprint arXiv:1511.06581, 2017.

[5] van Hasselt, H., Guez, A., & Silver, D. Deep Reinforcement Learning with Double Q-Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1)., 2016.

[6] Plappert M,Rychowicz M,RAY A,et al.Multi-goal reinforcement learning: Challenging robotics environments and request for research[J].arXiv preprint arXiv,2018(2):64.