# Asynchronous Deep Q-network in Continuous Environment Based on Prioritized Experience Replay

**Hongda Liu[a], Hanqi Zhang[b] and Linying Gong[c]**

College of Computer Science and Technology, Jilin University, Changchun 130012, China

[a]liuhd2116@mails.jlu.edu.cn, [b]zhanghq2116@mails.jlu.edu.cn, [c]gongly2116@mails.jlu.edu.cn

**Keywords:** Deep Q-network, Continuous Environment, Prioritized Experience Replay, Asynchronous

**Abstract:** Deep Q-network is a classical algorithm of reinforce learning, which is widely used and has many variants. The research content of this paper is to optimize and integrate some variant algorithms so that it has the advantage of running in the continuous environment, and improve the learning efficiency by Prioritized Experience Replay and multiple agents' asynchronous parallel method, and establish the asynchronous Deep Q-network framework based on priority Experience Replay in the continuous environment. This paper uses some games in the Atari 2600 domain to test our algorithm framework, which achieved good results, improved stability, convergence speed and improved performance.

## 1. Introduction

Deep reinforcement learning is one of the key research directions in the field of artificial intelligence. It has strong versatility and can be applied to many application fields, from end-to-end game control, intelligent robot control, facial recognition, and even natural language processing, those have a combination with deep learning. In recent years, there have been many research results, among which the Deep Q-network algorithm proposed by Googles Deep Mind team at NIPS in 2013 combines deep learning with reinforcement learning to achieve a new algorithm for end-to-end learning algorithm from perception to action [1]. Later, many studies refer to this algorithm and propose new improvement strategies to improve the learning efficiency of reinforcement learning.

After studying the classical algorithm Deep Q-network of deep reinforcement learning and various variants in recent years. This paper conducted an analysis and found that some problems still exist in the algorithm. This paper mainly improves from the following three aspects. Firstly, Deep Q-network needs to output value for each action. If it is a continuous action, the output value is not desirable, and it needs to be improved so that it can be learned in a continuous action environment. Secondly, the importance of different samples varies greatly. The method of random sampling does not make full use of the differences between samples. Moreover, the structure of convolutional neural networks is limited, and there is a strong correlation between experiences. It is not good to use long-term experience. Thirdly, the training time of Deep Q-network is very long. This article uses asynchronous parallelism to reduce training time. Therefore, after considering the stability and convergence of the algorithm, this paper proposed an algorithm framework to solve the above these problems one by one.

Firstly, the deterministic strategy is used to change the probability strategy and round update of policy gradient, and it is combined with Deep Q-network algorithm, so that Deep Q-network algorithm can be applied to continuous environment and then Temporal-Difference error (TD-error) is used to determine the priority based on the test result. Make the algorithm can learn more efficiently by using priority to determine the order of experience replay. Finally, consider using multiple agents to perform sampling learning independently, asynchronous parallelism, and improve the running speed of algorithm by optimizing the learned experience. Through the above improvements, the efficiency of the algorithm can be increased, the application range can be expanded, and the performance of the algorithm can be improved under the premise of satisfying the effective requirements, ensuring stability and convergence.

## 2. Related work

The Deep Q-network combines the Q-learning algorithm with the neural network Convolutional neural network (CNN), and inputs the state and action of the Q-learning algorithm into the neural network to obtain the Q-value of the action in the current state. In order to avoid the use of forms in the Q-learning algorithm to store the state, and the action corresponding to the state, it is impossible to solve the problem of high order of magnitude state calculation under complex problems.

Several variants of Deep Q-network algorithm produced by international researchers in recent years include Prioritized Experience Replay [2], Actor Critic, and A3C and so on. Prioritized Experience Replay, which changes the experience replay algorithm used by Deep Q-network, uses TD-error as a criterion to sample, learns more frequent conversions through replay, and expands the scope of application, making the algorithm equally efficient under special circumstances. The Actor-Critic [3] algorithm combines Deep Q-network with probability-based Policy Gradients algorithm, by using Policy Gradients as actors, and Deep Q-network as critic, letting actors select values first, then using critic to judge reward effects, and feedback affects actors. Update the parameters of the actor to make it more efficient. The Asynchronous Advantage Actor-Critic [4] algorithm is also based on Actor-Critic, which aims to solve the problem of Actor-Critic non-convergence. The difference is that it creates multiple parallel environments, allowing multiple agents with substructures to update the parameters in the main structure simultaneously in these parallel environments. The parameters of the main structure are not interfered with each other, and the parameter update of the main structure is interfered by the discontinuity of the sub-structure submission update, so the correlation of the update is reduced and the convergence is improved.

In the comprehensive comparison of the variants of the Deep Q-network algorithm, it is found that although the current algorithm has a great improvement in efficiency and convergence compared with the original Deep Q-network, in the improvement of the original Deep Q-network algorithm, the above mentioned problems are not solved uniformly. Therefore, this paper proposes a unified and reasonable solution to optimize the efficiency and convergence of the Deep Q-network algorithm by extending the Deep Q-network algorithm, and expand its application range.

## 3. Methods

In this part, the details of the algorithm framework will be detailed, mainly to solve the above problems, and to integrate and unify each other, which are summarized into three aspects. First, make Deep Q-network applicable to continuous environment. Second, adjust the parameters by prioritizing the experience to speed up the processing speed of Deep Q-network for special problems. Third, improve the efficiency of the algorithm by asynchronous parallelism and collaborative operation.

### 3.1 Continuous environment

Before extending Deep Q-network to a continuous environment, explain the basic idea of Deep Q-network, and then optimize and improve it.

Q-learning uses a form to store each state [5], and the Q-value represented by the action corresponding to the state can be applied well when the amount of data is small, but if the amount of data is large, a large amount of calculation will be generated. The efficiency is seriously reduced. The problems actually solved in reality are somewhat complicated, and the number of states may reach the order of magnitude that computer devices cannot test.

So Deep Q-network introduced a neural network, using state and action as input to the neural network. After the neural network forwardly propagated, the Q-value of the action in the current state is obtained, so that it is not necessary to record the Q-value in the table, but directly use Q-value generated by the neural network. There is also a form of inputting only the state value, outputting the respective Q-values of all the actions, and directly selecting the action having the maximum value as the next action to be performed. Neural network training is to optimize a loss function loss function. It is therefore necessary to have samples and then use the gradient descent method to update the parameters of the neural network by backpropagation. The update of the Q-value depends on the original Q-value and Reward:

$$R_{t+1} + \alpha max Q(S_{t+1}, a) \tag{1}$$

And because Deep Q-network takes this approach, it can only be applied in discrete environments. Deep Q-network gets the action value through the input of the state in a continuous environment, which is caused by the huge amount of data. Therefore, it is considered to improve it, retain the selection of the value, and increase the neural network that selects the action by probability on the basis of the original, and combine it to make it apply and continuous environment.

First, construct a network that selects actions by probability, input the state to calculate the probability of the action, and thus extract the action, so that the action can be selected by probability in a continuous interval. It is not enough to have this step alone. In the next step, Deep Q-network algorithm is integrated into the evaluation of the action. Through feedback on the experimental results, the reward value is obtained, the previous network is affected, the parameters are updated, and the learning efficiency of the whole algorithm is gradually improved. However, in the experiment, since the algorithm includes two neural networks and runs in a continuous environment, the parameters are updated in real time, so that there is correlation before and after the parameter update, which causes the neural network to consider the problem one-sided. When the test data is large, the learning efficiency of neural networks is affected. Therefore, further improvements are needed, and priority experience replay is introduced.

### 3.2 Priority experience replay

Under normal circumstances, experience replay saves the samples obtained by the system exploration environment, and then samples the samples to update the model parameters. It is similar to extract sample to update model parameters when the environment database is established. According to the above problem, the sampling needs to be improved. The probability of lowering the correlation makes the result of the algorithm more close to the target data, so the priority experience is used for replay, and TD-error is used as the criterion for sampling, and more frequent conversion is learned through replay.

The specific method is still to establish a sample database, but the change sampling method is not the original random sampling, but is based on the sample priority in Memory. This involves the determination of the sample priority value, how to prioritize. In this paper, TD-error is used, which is the difference value of reality of Q-value and estimate of Q-value to specify the degree of priority learning. If the TD-error is larger, there is still much room for improvement in the prediction accuracy, so the more the sample needs to be learned, the higher the priority. At the same time, according to the target results that need to be obtained, the priority level is adjusted to make it suitable for some experiments with low probability of success, and the probability of successful learning is expanded. Because TD-error has priority, it needs to be sampled according to priority in the experiment. If the sample is sorted every time, it will consume computational power. Therefore, the Sum Tree [6] structure is used, and it is not necessary to sort the samples every time. By prioritizing the experience replay, the algorithm can sample more efficiently, reduce the correlation, and can adjust the priority of the sample according to the specific problem, and get better learning speed in some experiments with low probability of success. The next step is to consider adding asynchronous parallelism to further improve the learning efficiency of the algorithm.

## 3.3 Asynchronous parallelism

The improved version of Deep Q-network mentioned above employs a prioritized experience replay mechanism to eliminate correlation between training data. However, experience replay requires a large amount of sample data to be accumulated first, and real-time interaction between the environment and the agent requires a lot of storage space and computing power, which reduces the efficiency of the algorithm to some extent. Therefore, it is considered to asynchronously parallel multiple agents, and the data is tested by running a multi-core processor of the type that is running at the same time, which saves running time and makes the algorithm more efficient.

The specific method is to first establish multiple agents, that is, multiple replicas with secondary structures, and measure them in the parallel environment created, and update the parameters in the main structure in these parallel environments. The parameter update of the main structure is interfered by the discontinuity of the sub-structure submission update. The different states of the multiple agent tests are used to eliminate the correlation between the state transition samples generated during the training process. The main structure at this time brings together the learning experience of each substructure, affecting the parallel substructures, updating their parameters, and getting the latest methods to make them equally efficient. The bias-variance trade-off method is used to help quickly spread the newly observed rewards to the old state and establish reliable real-time feedback. The algorithm as a whole does not need to be continuously updated as before, and the continuity is eliminated by the parallel sub-body, and the memory can be updated without using the previous memory.

## 4. Experimrnt

The algorithm in this article is debugged under Tensorflow and OpenAI gym. Some games in the Atari 2600 domain are used to test the algorithm effect. The following are the experimental results.

## 4.1 Continuous environment

According to the above method, a network is selected by probability to select the action, combined with the Deep Q-network algorithm, and the probability of the action is calculated by inputting the state, so that the action can be selected by probability in a continuous interval. At the same time, the reward and punishment value is set as the evaluation of the action, and it is fed back to the network to update the parameters. Here, Gym Cartpole is used here to test the algorithm.

Cartpole is a continuous motion test, we use algorithms to learn its game strategy to test the running effect in a continuous environment.

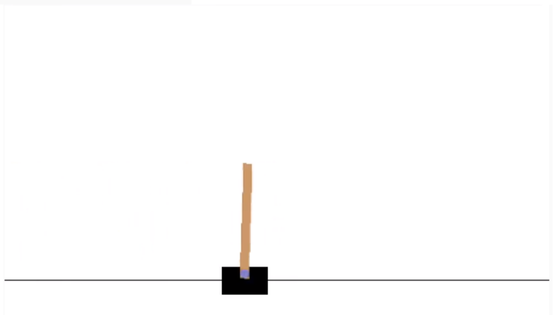

Figure 1. The algorithm runs successfully on Cartpole.

The Experiment tested 3000 rounds, and the learning effect was very good, and the improved Deep Q-network algorithm was extended to apply to the continuous environment. The following is a further experiment on the relevance of its data.

## 4.2 Priority experience replay

This paper implemented the natural DQN algorithm to make it suitable for continuous environments, then added priority experience playback, and TD-error calculation priorities.In order to meet the requirements of the method, we added the variable ISWeights, which is the Import-Sampling Weights, to restore the sampling probability distribution disturbed by the Prioritized replay. We simplified the calculation method and merged some steps.E.g:

$$w_j = (N \cdot P(j))^{-\beta} max_i w_i \tag{2}$$

$$prob = p/self.tree.total\_p \tag{3}$$

The corresponding memory used by DQN is also different, so the way of storing the transition changes.

The game we tested was MountainCar, and below is a comparison of the two.

It can be seen that dqn with experience playback requires less time to learn and has a faster learning speed.

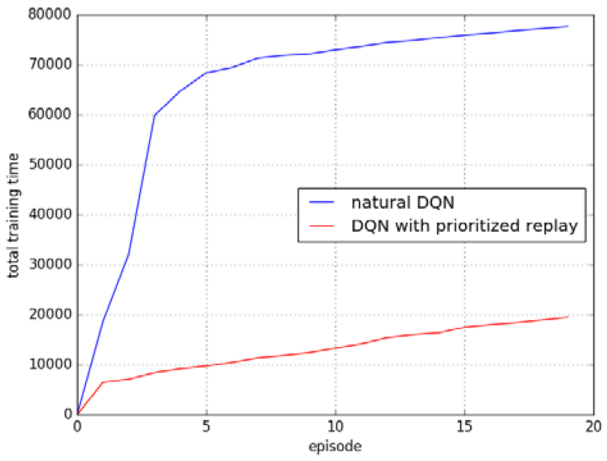

Figure 2. Compare the general Deep Q-network with the experimental results of Deep Q-network played back with prior experience.

## 4.3 Asynchronous parallelism

Further, in order to improve the operation efficiency, we asynchronously parallelize multiple agents and update the parameters in the main structure by running simultaneously. The parameters of the main structure are updated, which affects the parameter changes in the substructure. The deviation-variance trade-off method is used to establish reliable real-time feedback and improve the efficiency of the algorithm.

In the experiment we built 4 workers and there is also a main program. The main program has global net and its parameters, each worker has a copy of global net named local net, and you can periodically push updates to global net, and then get a comprehensive update from global net. The experiment set the push and pull functions. If call pull in sync, the current worker will get the latest parameters from global net. If push the push in sync, it will upload the update to global net. The experiment use the Normal distribution to select the action, so while building the neural network, we output the mean and variance of the action, then put it into the Normal distribution to select the action. When calculating the loss, we also need to use the previously mentioned TD-error as the gradient. Ascent's orientation.

Here is the game Cartpole, the following is the experimental effect.

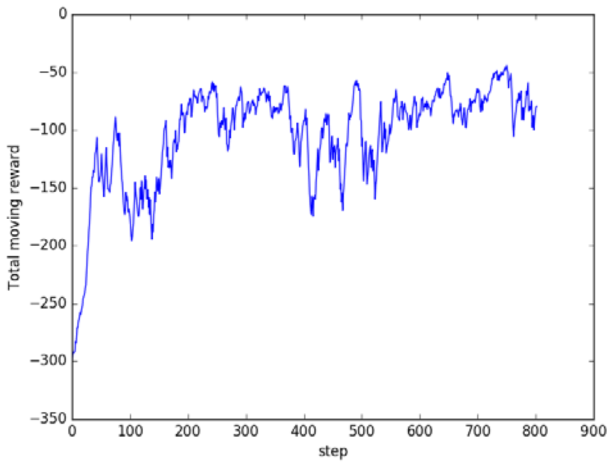

Figure 3. The experimental effect of asynchronous DQN is adopted.

The learning result is judged by the reward of the moving average on the left side. It can be seen that the parallel algorithm has a good running efficiency and achieves the experimental purpose.

## 5. Conclusion

In this paper, we present an improved algorithm for Deep Q-network. Mainly improved three aspects. First, the use of the deterministic strategy of probability combined with the neural network of Deep Q-network, so that the Deep Q-network algorithm can be applied to the continuous environment. Second, the use of priority experience playback, using TD-error and test results to determine the sample Experience priority, adjust relevant parameters, reduce correlation. Third, through asynchronous parallel multiple agents, simultaneous independent sampling learning, interact with the main body to improve algorithm efficiency. The experiment has proved that the improved algorithm has found a better strategy, and the performance is much better than the initial Deep Q-network. It has achieved good results in the Atari 2600 field. In future, we will continue to expand the considerations, from three gradually increasing, expecting Deep Q-network algorithm to have better performance improvement.

## References

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning [J]. Computer Science, 2013.

[2] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver.Prioritized experience replay. In: International Conference on Learning Representations. Puerto Rico, 2016. https://arxiv. org/pdf/ 1511.05952.pdf

[3] Peters J, Schaal S. Natural Actor-Critic [J]. Neurocomputing, 2008, 71 (7-9): 1180-1190.

[4] Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, Sergey Levine. Continuous deep Q-learning with model-based acceleration. In: Proceedings of the 32nd International Conference on Machine Learning, 2016, 2829-2838.

[5] Watkins C J C H, Dayan P. Technical Note: Q-Learning [J]. Machine Learning, 1992, 8 (3-4): 279-292.

[6] Schaul T, Quan J, Antonoglou I, et al. Prioritized Experience Replay [J]. ComputerScience, 2015.