Abstract: The sample-based prioritised experience replay proposed in this study is aimed at how to select samples to the experience replay, which improves the training speed and increases the reward return. In traditional deep Q-networks (DQNs), it is subjected to random pickup of samples into the experience replay. However, the effect of each sample is different for the training process of an agent. A better sampling method will make the agent training more effective. Therefore, when selecting a sample to the experience replay, the authors first allow the agent to learn randomly through the sample optimisation network, and take the average value returned after each study, so that the mean value is used as a threshold for selecting samples to the experience replay. Second, on the basis of sample optimisation, the authors increase the priority update and use the idea of reward-shaping to give additional reward values to the returns of certain samples, which speeds up the agent training. Compared with traditional DQN and the prioritised experience replay DQN, this study uses OpenAI Gym as a platform to improve agent learning efficiency.

1 Introduction

Today, as the concept of artificial intelligence is becoming more and more popular, the algorithm model of deep reinforcement learning, which is a research hotspot of artificial intelligence, has been more and more popular. The deep reinforcement learning algorithm proposed by Mnih et al. [1], deep reinforcement learning has been shown to be comparable to human players and even surpasses human players in dealing with complex problems such as Atari 2600 games that tend to be real. In addition, deep reinforcement learning has achieved significant success in various tasks, such as robotic autonomous navigation and path planning (Kober and Peters [2], Mirowski et al. [3], Cui et al. [4]), unmanned technology [5], natural language processing [6, 7] and so on. In 2016, the appearance of AlphaGo [8] made an uproar in the field of Go, defeating the Korean Go player Sedol Lee with a long training time. In 2016, with the proposed of double DQN [10], Dueling DQN [11] and priority experience replay [12] algorithm, the training speed and reward of the agent gradually increased. Double DQN optimises the objective function based on DQN, so that the value function of selecting the action value is different from the value function used to evaluate the action value. This reduces the overestimation of the sample, avoids the generation of sub-optimal strategies, and also greatly improves the performance of the DQN. The Dueling DQN divides the action value function into two parts such as the state value function and the advantage value function. In the course of the experiment, when redundant or similar actions occur, thanks to the separation of the value function and the advantage function, the agent can quickly learn the value of these new actions without having to relearn the value of the state. In this way, the training speed of the agent is further improved, and the training performance is better. In the article of priority experience replay, based on the previous improvements, the sample of the experience replay buffer is prioritised, which makes it easier to select larger samples for the training of the agent, shortening the training time and improving the learning efficiency. In addition, Osband et al. [13] proposed a better exploration strategy based on DQN. O'Donoghue et al. [14] proposed Q-learning based on policy gradient. These have further improved the efficiency of intelligent learning.

The algorithm structure proposed above accelerates the training of the agent and improves the learning effect. However, for complex models and learning environments, when the storage capacity of the experience replay buffer reaches 100,000 or more, the training time of the agent is still longer. In 2016, Google's DeepMind's prioritised experience replay proposed two options for the use of experience replay buffer: one is to choose which experience to store; the second is which experience to be replay. The prioritised experience replay mainly solves the second problem, which is how to prioritise the sample tracks stored in the experience replay buffer. By prioritising the temporal-difference (TD) error, the selected probability that the sample which is more effective for the agent learning is increased, and the probability that the important experience is played back is increased, so that the training result of the agent is more effective and a better return is obtained. In this paper, it is proposed that the prioritised experience replay based on sample optimisation model is focused on solving which experiences are selected for replay. Prioritised experience replay based on sample selects before putting the experience into the experience replay buffer. The more effective and important experience of the agent training is stored in the experience replay buffer as a later experience replay. In the training process after the training, since the samples in the experience replay buffer are selected, the training process is more...
Fig. 1 Structure diagram of prioritised experience replay based on sample optimisation

efficient. Secondly, this paper also updates the priority of the experience sample. Also the priority of the samples in prioritised experience replay is not updated in real time. In this paper, when each transition is sampled, the priority corresponding to each sample and the TD error are updated. Thus, during the training process, the priority of the sample for different environmental states is different. This helps the agent to train more realistically and improve training efficiency. In addition, drawing on the idea of reward-shaping [15], an additional reward is given to the first few state samples reaching the final state, so that the agent can learn the final state more quickly during the training process.

2 Prioritised experience replay based on sample optimisation

2.1 Network structure diagram

Fig. 1 is the structure diagram of prioritised experience replay based on sample optimisation. In the traditional experience replay buffer, sampling is randomly selected for the sample, and there is no effective selection for the sample to enter the experience replay buffer. This paper proposes that before the sample enters the experience replay buffer, a training process is performed in advance, and a threshold of reward is obtained through the training process. When the return of the sample is greater than this threshold, the sample is placed in the experience replay buffer, and when the return of the sample is less than this threshold, the sample is discarded.

In the article of prioritised experience replay, a priority-based ranking of the samples of the experience replay buffer is proposed. When the absolute value of the TD error of each sample is large, if the priority of the sample is higher, the probability that the sample will be selected is larger. In this section, we add updates based on TD error priorities and reward-shaping of some samples. When the sample is selected, the TD error and priority of the sample are updated at the same time, so that the samples that are more effective training for the agent are more likely to be selected, and the effect of the agent training is accelerated.

2.2 Prioritised experience replay based on sample optimisation

2.2.1 Prioritised experience replay: This paper adopts the double DQN network structure. Different from the traditional single DQN network, it not only uses the convolutional neural network to approximate the behaviour value function, but also uses the target Q network to update the target. Also in the loss function, the max operation is not used to obtain the optimal value, but is used to obtain the action value of the optimal strategy. The same value is used to select and measure an action compared to the maximum Q value operation in DQN, which avoids the overly optimistic value estimate caused by the excessively high estimate of Double DQN. The loss function for this network is

$$L(\omega) = (r + \gamma Q(s', \alpha, \arg\max Q(s', a, \omega), a') - Q(s, a, \omega))^2$$

(1)

By calculating the gradient of w about the loss function, we can get the function

$$\partial L(\omega)/\partial \omega = E[(r + \gamma Q(s', \alpha, \arg\max Q(s', a, \omega), a') - Q(s, a, \omega))Q(s, a, \omega)]$$

(2)

where the parameter \(\alpha\) is used to select the action, which is from the current q network; \(a'\) is the parameter used to evaluate the action, which is from the target q network. The current q network selects the action, and the target q network calculates the q value.

The main function of the experience replay in the double DQN network is to solve the non-static distribution problem. The specific method is to store the samples obtained by interacting each time step agent with the environment into the experience replay buffer, and randomly pick out small batch samples for training during the training of the agent body and makes the training more efficient.

The use of experience replay will enable the agent to remember and reuse past experience. In the traditional DQN network structure, the past experience was obtained by uniform sampling, which would make all the experience in the experience replay buffer have the same probability, ignoring the importance of different experiences. Therefore, prioritised experience replay uses TD error to mark the importance level of each experience, so that the probability of important experience being sampled is greater, which greatly increases the efficiency of training.

In order to ensure sampling efficiency, a random sampling method was introduced. This method is an algorithm between the pure greedy prioritisation algorithm and the original uniform random sampling algorithm. It not only solves the problem that each sample is evenly extracted, but also solves the problem that a small TD error transition that was first seen in greedy prioritisation is stored in the experience replay buffer and will not be sampled for a long time. In addition, it also solves the problem of over-fitting. The random sampling method can ensure that the probability that the transition which is extracted is monotonous according to the priority. At the same time, it is guaranteed that zero probability extraction may not occur for low priority transitions. The transition with the label \(i\) is defined as

$$P(i) = \frac{p_i}{\sum_k p_k}$$

(3)

where \(p_i\) is the priority of the transition labelled \(i\), all of \(p_i\) are greater than zero, and \(a\) is the influence value that determines the importance of the sampling weight. If \(a = 0\), the importance sampling is not used, which is equivalent to uniform random sampling.

In this paper, in order to make the sorting network more robust, let \(p_i = 1/\text{rank}(i)\), where \(\text{rank}(i)\) is sorted by the size of \(k_i\). In (3), \(p\) is monotonous with respect to \(\|k_i\|\), and because of the ordering, it is not very sensitive to outliers, which enhances the stability of the system.

2.2.2 Sample optimisation: The algorithm proposed is to use the Double DQN network to randomly select samples for pre-training and obtain the corresponding average return before selecting samples to enter the experience pool, and put the samples with higher average returns into the experience pool. Then it will calculate the average of the final return value of the selected sample as the criterion for selecting whether to put the sample into the experience replay buffer. Then, the sample with better return value is put into the experience replay buffer, and the training is...
played with priority experience. Finally, the final return of the algorithm is improved, and the algorithm is optimised.

Compared to the original prioritised experience replay, this algorithm enhances the selection of samples into the experience replay buffer. Before selecting samples, the sample optimisation network uses the natural DQN algorithm to randomly select samples for training. The return value for each training is defined as

$$R = \sum_{i} r_i$$  \hspace{1cm} (4)$$

In this paper, the mean square error of $n$ samples is taken as the threshold for selecting and adding samples into the experience replay buffer:

$$R_{MSD} = \frac{1}{n} \sum_{i=1}^{n} (R_i - \bar{R})^2$$  \hspace{1cm} (5)$$

where $\bar{R}$ is the average of the return values of $n$ samples. If the final return value of the sample is $> R_{MSD}$, the sample is placed in the experience replay buffer, and if it is $< R_{MSD}$, the sample is discarded.

The set of all samples entering the experience replay buffer is defined as

$$S_{SAMPLE} = \sum M_i$$  \hspace{1cm} (6)$$

where $S_{SAMPLE}$ represents all samples entering the experience replay buffer and $M_i$ is the sample trajectory where the final return $R_i$ is $> R_{MSD}$.

However, in the training of the agent, it is necessary to explore some situations where the current performance is not good. For some samples with poor current returns, some of the samples should be put into the experience replay buffer, so the improvement of the $\xi$-greedy strategy is added when selecting samples. The improved sample set in experience replay buffer is defined as

$$S_{SAMPLE} = \begin{cases} \sum_{i} M_i + \sum_{j} M_j, & b \leq \xi \\ \sum_{i} M_i, & b > \xi \end{cases}$$  \hspace{1cm} (7)$$

where $M_j$ is a randomly selected sample track and $b$ is a random number in the $(0, 1)$ interval.

### 2.2.3 Priority update: The samples are prioritised based on priority according to the TD error during sample storage. Each time a sample is stored, a priority is assigned based on the absolute value of the TD error, and this data pair is placed in the experience replay buffer. Each time a sample enters the experience replay buffer, the priority of each sample is fixed and does not change with the training process of the agent. However, in the training process of the agent, as the training level is deepened, the actual priority of each sample has changed due to the change of the current TD error, so the probability of being selected should also change.

Therefore, this paper proposes a tracking update of the priority. When the sample is sampled once, its corresponding priority should also be updated. The corresponding priority is compared with the current TD error and the average TD error of the previously stored samples. The average TD error formula is defined as

$$\hat{\delta} = \frac{1}{k} \sum_{i} |e_i|$$  \hspace{1cm} (8)$$

The sample priority and TD error mean update formula is defined as

$$p_i' = p_i + \lambda (|\delta_i| - \hat{\delta})$$  \hspace{1cm} (9)$$

$$\delta_i' = \hat{\delta} + \gamma (|\delta_i| - \delta_i')$$  \hspace{1cm} (10)$$

where $p_i'$ is the sample priority after the update, $\delta_i'$ is the average TD error after the update, and $\lambda$, $\gamma$ are two positive parameters belonging to the $(0, 1)$ interval. The idea of this update formula is that a high TD error should increase the probability that the action will be selected again, while a low TD error should reduce this probability.

In order to improve the probability that the high priority samples are selected, this paper proposes a variant of the sample selection probability calculation, and makes the following modifications to the formula (3):

$$P(i) = \frac{\alpha p_i^{\beta/\alpha}}{\sum_{i} \alpha p_i^{\beta/\alpha}}$$  \hspace{1cm} (11)$$

where $\alpha$ is still a positive parameter. When $\alpha$ is a sufficiently large value, the picking probability of the sample is equivalent to a uniform random selection, it means that importance sampling is not used, which is equivalent to uniform random sampling.

In this algorithm, we not only preserve the priority of the sample in the prioritised experience replay, but also improve the training efficiency through sample selection and priority update. Sample optimisation DQN with prioritised experience replay algorithm is shown in Fig. 2.

#### 2.2.4 Partial reward-shaping:

In most models of deep reinforcement learning, the external rewards that an agent obtains in the environment are usually sparse. That is, when the final goal is learned, a return value other than zero is returned, and the returns obtained by other states are usually zero. Therefore, in order to speed up the training of the agent, this article adds an internal reward to the return of the first $n$-step state to the final goal, increasing their total return.

For the $Q$ value of each step in the training, the Bellman formula shows

$$Q'(s, a) = E_{\pi}[r_{t+1} + \gamma \sum_{k=0}^{n} r_{t+k+1} | s_t = s, a_t = a]$$

After adding an internal reward, the $Q$ value of each step is defined as

$$Q'(s, a) = \sum_{s'} P_{a,s'} [r'(s, a, s') + \gamma Q'(s', a')]$$  \hspace{1cm} (13)$$

where $r'(s, a, s')$ represents the total reward, $r(s, a, s')$ represents the external reward, and $F(s, a, s')$ represents the internal reward. In this article, the internal rewards for the first $n$ steps of the final goal are defined as

$$F(s, a, s', n) = \kappa^n r_i$$  \hspace{1cm} (15)$$

where $\kappa$ is a constant and $r_i$ is the internal return value of the final state. By increasing the internal reward, this algorithm increases the sampling probability of the agent's first $n$-step state of the final target in training. So that the agent can learn the final goal faster. This method not only speeds up the time for the agent to learn the final state, but also improves the training efficiency. The algorithm of partial reward-shaping is shown in Fig. 3.
Results

We use Gym and Atari [16] as environment platforms for comparison experiments, which proves the superiority of the proposed prioritised experience replay based on sample optimisation.

In this paper, MountainCar-v0, Acrobot-v1 in Gym and Breakout-v0, Riverraid-v0 in Atari are selected as experimental environments. The traditional DQN, priority experience replay DQN, and prioritised experience replay based on sample optimisation algorithm are used for experiments. The good performance of the sample optimisation prioritised experience replay algorithm is demonstrated by a comparison of total returns.

For the two Gym environments Acrobot-v1 and MountainCar-v0 in Fig. 4, the experimental setup is as follows: the training network is a three-layer fully connected network. Each layer has 64 neurons. The actions of the two environments are 2 and 3, which correspond to the number of tasks in their respective environments. The activation functions corresponding to the layer network are relu, relu, and softmax. In the prioritised experience replay based on sample optimisation, the other parameters are consistent with the parameters in the prioritised experience replay, and the experimental settings are also consistent.

For the Riverraid and Breakout environments in Atari shown in Fig. 4, the experimental setup is as follows: the training network is a five-layer network, and the first three layers are convolutional neural networks. The first layer of convolution kernels is 8 × 8, and the number of channels is 4. The step size is 4; the second layer convolution kernel is 4 × 4, the channel number is 32, and the step size is 2; the third layer convolution kernel is 3 × 3, the channel number is 64, and the step size is 1. The latter two layers of the network are fully connected networks, the number of neurons is 512, and the number of corresponding tasks is the number of actions. The activation functions corresponding to the five-layer network are relu, relu, relu, relu, and softmax. In the prioritised experience replay based on sample optimisation, the other parameters are consistent with the parameters in the prioritised experience replay, and the experimental settings are also consistent.

Fig. 5 shows the return curves of the three algorithms over four discrete motion spaces. In Acrobot-v1 and MountainCar-v0, prioritised experience replay based on sample optimisation can quickly learn the final goal and return better than the original DQN; however, compared to the prioritised experience replay DQN, the return curve between them is not much different. The reason may be that the Acrobot-v1 and MountainCar-v0 environments have relatively simple action spaces and fewer states. In the Acrobot-v1, only the agent can get a return when it exceeds the black line above. In the MountainCar-v0, the effective return can only be obtained when the car reaches the top of the mountain. So, in these two environments, the sample has fewer states and the reward return is sparser. However, the prioritised experience replay based on sample optimisation is mainly optimised for the sample to improve the reward of the agent, so the return is improved compared with the prioritised experience replay DQN, but the improvement is not much. In the two Atari game environments of Riverraid-v0 and Breakout-v0, the prioritised experience replay based on sample optimisation is better than the traditional DQN on sample optimisation, the other parameters are consistent with the parameters in the prioritised experience replay, and the experimental settings are also consistent.

For the Riverraid and Breakout environments in Atari shown in Fig. 4, the experimental setup is as follows: the training network is a five-layer network, and the first three layers are convolutional neural networks. The first layer of convolution kernels is 8 × 8, and the number of channels is 4. The step size is 4; the second layer convolution kernel is 4 × 4, the channel number is 32, and the step size is 2; the third layer convolution kernel is 3 × 3, the channel number is 64, and the step size is 1. The latter two layers of the network are fully connected networks, the number of neurons is 512, and the number of corresponding tasks is the number of actions. The activation functions corresponding to the five-layer network are relu, relu, relu, relu, and softmax. In the prioritised experience replay based on sample optimisation, the other parameters are consistent with the parameters in the prioritised experience replay, and the experimental settings are also consistent.
and prioritised experience replay DQN. The return curve is higher, and the ultimate goal can be learned faster. In the Riverraid-v0, the agent gains scores by controlling the actions of its own aircraft, killing or avoiding enemies. When hit by the enemy, the game state, which increases sampling probability of the internal reward is added to the first n-step samples of the final state. In Gym and Atari environment platform, by comparing prioritised experience replay DQN with traditional DQN, the prioritised experience replay based on sample optimisation improves algorithm effect and accelerates the learning efficiency of the agent.

5 Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 61976215 and Grant 61772532.

6 References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., et al.: ‘Human-level control through deep reinforcement learning’, Nature, 2015, 518, (7540), pp. 529–533
[2] Kober, J., Peters, J.: ‘Reinforcement learning in robotics: a survey’, Int. J. Robotics Res., 2013, 32, (11), pp. 1238–1274
[3] Mirowski, P., Pascanu, R., Viola, F., et al.: ‘Learning to navigate in complex environments’. Proc. of Int. Conf. on Learning Representations (ICLR), San Juan, Puerto Rico, April 2017
[4] Cui, R., Yang, C., Li, Y., et al.: ‘Adaptive neural network control of UAVs with control input nonlinearities using reinforcement learning’, IEEE Trans. Syst. Man Cybern.: Syst., 2017, 47, (6), pp. 1019–1029
[5] Sallab, A., Abdou, M., Perot, E., et al.: ‘Deep reinforcement learning framework for autonomous driving’, Proc. of Int. Conf. on Neural Information Processing Systems Workshop-MLITS, Barcelona, Spain, December 2016, pp. 70–76
[6] Li, J., Monrose, W., Ritter, A., et al.: ‘Deep reinforcement learning for dialogue generation’, Proc. of Conf. on Empirical Methods in Natural Language Processing, Austin, TX, USA, November 2016, pp. 1192–1202
[7] Cuayahuitl, H., Yu, S., Williamson, A., et al.: ‘Scaling up deep reinforcement learning for multi-domain dialogue systems’. Proc. of Int. Joint Conf. on Neural Networks, Anchorage, AK, USA, May 2017, pp. 3339–3346
[8] Silver, D., Huang, A., Maddison, C.J., et al.: ‘Mastering the game of go with deep neural networks and tree search’, Nature, 2016, 529, (7587), pp. 484–489
[9] Fu, J., Levine, S., Abbeel, P.: ‘One-shot learning of manipulation skills with online dynamics adaptation and neural network priors’. Computer Science, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Daejeon, Korea, October 2016, pp. 4019–4026
[10] Van Hasselt, H., Guez, A., Silver, D.: ‘Deep reinforcement learning with double q-learning’. Proc. of AAAI Conf. on Artificial Intelligence (AAAI), Phoenix, AZ, USA, February 2016, pp. 2094–2100
[11] Wang, Z., Schaul, T., Hessel, M., et al.: ‘Dueling network architectures for deep reinforcement learning’. Proc. of Int. Conf. on Machine Learning (ICML), New York, NY, USA, June 2016, pp. 1995–2003
[12] Schaul, T., Quan, J., Silver, D., et al.: ‘Prioritized experience replay’. Proc. of Int. Conf. on Learning Representations (ICLR), San Juan, Puerto Rico, May 2016
[13] Osband, I., Blundell, C., Pritzel, A., et al.: ‘Deep exploration via bootstrapped DQN’. Proc. of Annual Conf. on Neural Information Processing Systems (NIPS), Barcelona, Spain, March 2016, pp. 4026–4034
[14] O’Donoghue, B., Munos, R., Kavukcuoglu, K., et al.: ‘PGQ: combining policy gradient and q-learning’. Proc. of Int. Conf. Learning Representations, France, April 2017, pp. 1–15
[15] Ng, A.Y., Harada, D., Russell, S.: ‘Policy in variance under reward transformations: theory and application to reward shaping’. Proc. of Int. Conf. on Machine Learning (ICML), Blud, Slovenia, June, 1999, 99, pp. 278–287
[16] Bellemare, M.G., Naddaf, Y., Veness, J., et al.: ‘The arcade learning environment: an evaluation platform for general agents’. J. Artif. Intell. Res., 2012, 47, (1), pp. 253–279

Fig. 5 Return curves
(a) Acrobot-v1, (b) MountainCar-v0, (c) Riverraid-v0, (d) Breakout-v0