Journal of Computers Vol. 34 No. 2, April 2023, pp. 41-51
doi: 10.53106/199115992023043402004

A Deep Reinforcement Learning-Based Approach in Porker Game

Yan Kong[1], Yefeng Rui[1], Chih-Hsien Hsia[2,3*]

[1] School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, China
[2] Department of Computer Science and Information Engineering, National Ilan University, Yilan, Taiwan, R.O.C
[3] Department of Business Administration, Chaoyang University of Technology, Taichung, Taiwan, R.O.C
{rui_yefeng, kongyan4282}@163.com, chhsia625@gmail.com

Received 17 June 2022; Revised 31 July 2022; Accepted 3 August 2022

Abstract: Recent years have witnessed the big success deep reinforcement learning achieved in the domain of card and board games, such as Go, chess and Texas Hold'em poker. However, Dou Di Zhu, a traditional Chinese card game, is still a challenging task for deep reinforcement learning methods due to the enormous action space and the sparse and delayed reward of each action from the environment. Basic reinforcement learning algorithms are more effective in the simple environments which have small action spaces and valuable and concrete reward functions, and unfortunately, are shown not be able to deal with Dou Di Zhu satisfactorily. This work introduces an approach named Two-steps Q-Network based on DQN to playing Dou Di Zhu, which compresses the huge action space through dividing it into two parts according to the rules of Dou Di Zhu and fills in the sparse rewards using inverse reinforcement learning (IRL) through abstracting the reward function from experts' demonstrations. It is illustrated by the experiments that two-steps Q-network gains great advancements compared with DQN used in Dou Di Zhu.

Keywords: deep reinforcement learning, artificial intelligence, Porker Game, sparse reward

1 Introduction

Games, in which agents can only receive partial observation from the environment, are defined as imperfect information games and the rest are recognized as perfect information games. For example, Go is a typical perfect information game while Texas Poker and Dou Di Zhu are imperfect information games. In recent years, artificial intelligence (AI) techniques have emerged in multitude in popular games and achieved satisfactory performance. AlphaGo [1] is the first AI agent that defeated the world's top GO player [2] by self-play without human knowledge and the following researches including AlphaGo Zero [3], AlphaZero [4] and MuZero [5] have gained superhuman performance in the game of GO as well. Libratus [6], DeepStack [7] and AlphaHoldem [8] have proved to be great success in Texas Hold'em Poker. According to these, reinforcement learning (RL) [9] may be a powerful solution for gaming [10].

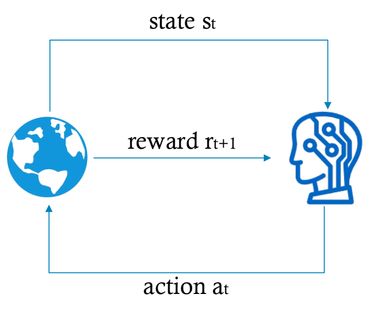


Fig. 1. The process of reinforcement learning

* Corresponding Author

Dou Di Zhu could be modeled into a Markov decision process (MDP), as shown in Fig. 1, thus RL has become one of the widely used AI algorithms in Dou Di Zhu. In an MDP, an agent interacts with the environment, in specific, at moment , the agent comes to state , takes action , and receives a reward from the environment and iterates till arriving the final state. When it comes to introducing RL with deep learning methods (DRL) into Dou Di Zhu, we are confronted with several challenges. The first one is the enormous and complicated action space. There are 54 cards in Dou Di Zhu which can be combined with each other according to the rules and the number of legal combinations of cards (*i.e.*, legal actions) reaches 27,472 consequently. RL algorithm (*e.g.*, DQN [11]) are problematic in very large action space due to overestimating issue [12], because approximating the maximum action value is difficult to control when using function approximation [13] and get more difficult with the increasement of the action space. To reduce the number of possible actions, Combinational Q-learning (CQL) [14] decouples the actions into two parts: decomposition selection and final selection of actions. However, decomposition of cards is extremely slow. Similarly, DeltaDou [15] pre-trains a kicker network based on heuristic rules to compress the action space. The second challenge is the sparse reward. In Dou Di Zhu, the information related to the reward provided by the environment is the final result of one round game, consequently, the reward an agent receives at each step is always zero and thus sparse. RL is a machine learning method with a reward mechanism [16], and as we know, sparse and delayed reward will make it hard to train the RL agent, slow down the training process and do harm to the performance of the Q-network. Against this, DouZero [17] enhances the traditional monte-carlo (MC) methods with deep neural network (DNN) to make effective use of the final results of games, but meanwhile it introduces another problem, that is, it needs an enormous number of integrated trajectories.

Against the above background, this paper introduces an approach named Two-steps Q-network which is based on traditional deep reinforcement learning algorithm DQN, to playing Dou Di Zhu. Firstly, we compress the huge action space of Dou Di Zhu by means of splitting the whole action space into two parts: the action space of main-groups and the one of kicker-cards, according to which the main-group Q-network and the kicker-card Q-network are constructed. For example,3334, an action in normal action space introduced in other works, is split into 333* in main-group action space and ***4 in kicker-card action space. Main-group and kicker-card Q-Networks are responsible for different parts. Secondly, in order to solve the problem of sparse reward, reward functions are constructed for both main-group and kicker-card Q-networks. Specifically, we utilize inverse reinforcement learning (IRL) [18] to learn three reward functions for the three different roles (i.e. the landlord, the down-peasant and the up-peasant) respectively in the main-group Q-network, and reward shaping is adopted in kicker-card Q-network. The main contributions of this work include: 1) This work splits the whole action space of Dou Di Zhu to compress the action space through defining new action spaces for main-groups and kicker-cards respectively. 2) IRL and modified prioritized experience replay are utilized to address the problem of sparse reward in Dou Di Zhu.

2 Background

2.1 Dou Di Zhu

Dou Di Zhu (fighting the landlord) is a traditional Chinese poker game. Standard Dou Di Zhu game is composed of three players: one landlord and two peasants. The landlord plays against the two peasants. The initial deck used in Dou Di Zhu consists of 54 cards including 52 standard cards and 2 jokers. The ranks of the cards decrease progressively as follows: R (Red Joker), B (Black Joker), 2, A (Ace), K (King), Q (Queen), J (Jack), 10, 9, 8, 7, 6, 5, 4, 3. Each rank of standard cards has 4 single ones of different suits (i.e., spade, heart, diamond and club) which have equivalent values in the same rank.

At the beginning of each round, one of the players shuffles the cards and deals 17 cards to each player. Then, players decide whether to bid for being the landlord according to their starting hands. The remaining three leftover wild cards will be dealt to the landlord after the landlord is decided. The landlord has priority to playing cards of any legal combinations at the beginning. The next player should choose to pass (skip playing cards) or play cards which have higher rank according to the rules. If the landlord plays out of all his cards first, the landlord wins the game, otherwise, peasants win. Several definitions used in this work is listed as follows:

Type: the legal combinations of card(s) which is allowed in Dou Di Zhu, and there are 14 types in Dou Di Zhu.

Main-group: The combination of card(s) which can decide the rank. Taking a combination of cards- 3334 as an example, 333 is the main-group. In some cases, one main-group is consisted of some single cards which have the same rank. The same rank shared among the single cards in main-group decide the rank of the main-group.

Kicker-card: The combination of card(s) which meets the requirement of the type and is not related to the rank. Taking a combination of cards - 3334 as an example, 4 is the kicker-card. As the fact that kicker-card is not related to the rank of the combination, 3334 and 3335 have the same rank in Dou Di Zhu.

Down-peasant: The peasant player who plays after the landlord.

Up-peasant: The peasant player who plays after the down-peasant.

2.2 Deep Q-Learning

Q-learning [19] is a traditional value-based RL algorithm, the core of which is as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha(r+\gamma \max_{a'} Q(s',a')), \tag{1}$$

where α denotes the learning rate, γ is the discount factor and r is the immediate reward from the environment. Agents take actions under the guide of Q-values coupled with certain polices. Instead of using tabular storage to present and update Q-values, deep Q-learning utilizes a DNN to produce Q-values in order to make Q-values also applicable to environment which has complicated and huge state or action space. Deep Q-networks (DQN) are an extension of the Q-learning algorithm [20]. DQN adopts a replay memory to store the transition samples each of which is a four-tuple <s, a, r, s'>which means when an agent is at state , it takes action a, then it gains the reward r from the environment and comes to a new state s'. Q-network is trained by sampling a fixed batches of samples from the replay memory and is optimized through diminishing the square TD-ERROR δ :

$$\delta = r+\gamma \max_{a'} Q(s',a';\theta')-Q(s,a;\theta), \tag{2}$$

where θ $p_i^{'} = p_i + \eta$, and θ' are the parameters of the evaluate and target networks respectively. θ' denotes copied from θ periodically, and the loss function of target network is:

$$loss = \delta^2 \tag{3}$$

2.3 Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is an effective way to build reward functions. Many sequential decision-making problems can be tackled by IRL [21]. The core assumption of IRL is that experts (high-level players in real world) know the reward functions which can lead to optimal or approximately optimal strategy. IRL is useful to improve the performance of RL of which rewards are hard to be abstracted or even are not defined from the environment because collecting demonstrations is always easier than designing reward functions which are hard to be accurately generalized in games. There are two widely used approaches to constructing reward functions in IRL. The first one is behavioral cloning [22], based on supervised learning, in which agents directly copy the experts' action at the same state. The limitation of behavioral cloning is obvious due to the uncertain environments. The other approach is apprenticeship learning (AL) [23], which tries to find an optimal or approximately optimal policy through experts' demonstrations.

2.4 Prioritized Experience Replay

Experience replay [24] which follows the independent and identically distributed assumption and breaks the temporal correlations among the transitions has been proven to be successful in DQN. Prioritized experience replay (PER) [25] takes the lead in putting forward priorities for transitions. The core idea of PER is to more frequently replay experiences associated with successful attempts or extremely awful performance [26] through giving higher priorities to such experience transitions. TD-ERROR (as listed in Eq. (2)) is adopted to measure the significance of samples in PER, and the probability of sampling transition is defined as:

$$P(i) = \frac{p_i^\alpha}{\sum k p_k^\alpha}, \tag{4}$$

where α determines the degree of the priority to be utilized, and in case of $\alpha = 0$, prioritized experience replay is turned into the uniform case. p_i means the priority of the transition *i*, and one of the widely used way to obtain p_i is $p_i = |\delta i| + \varepsilon$, where ε is a positive constant to avoid the case that transition *i* is never sampled when δi is zero.

3 Two-steps Q-Network

In Fig. 2, we introduce our proposed framework two-steps Q-network, the states and actions encoding is specifically introduced in Sections 3.1, and 3.2 describes the structure of two-steps Q-network in details.

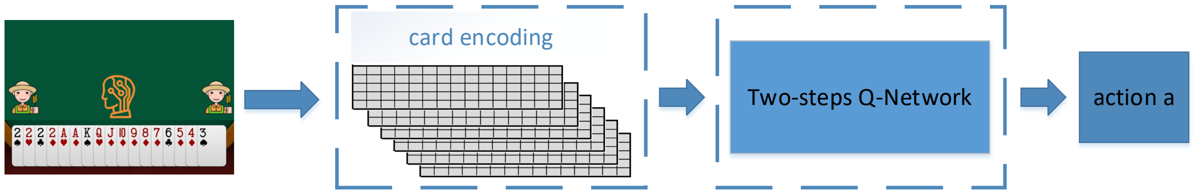
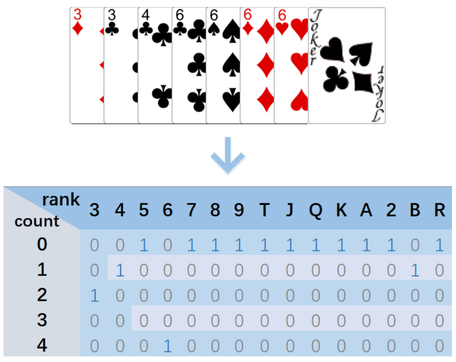


Fig. 2. The process of two-steps Q-network

3.1 Card Encoding

The information of a combination of cards is encoded into a 5×15 one-hot matrix, as shown in Fig. 3, in which columns correspond to the 15 ranks, and each row corresponds to the number of cards from 0 to 4. In specific, the information of the current hand, the union of the other two players' hands, the recent three actions (pass is not included) and the union of all cards which have been played are encoded into 5×15 matrices sequentially and concatenated together to form a 6×5×15 matrix in order to represents the information of one state.



count \ rank	3	4	5	6	7	8	9	T	J	Q	K	A	2	B	R
0	0	0	1	0	1	1	1	1	1	1	1	1	1	0	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Fig. 3. Encoding of cards

3.2 Structure of Two-steps Q-Network

RLcard [27] adopts a DQN to produce the Q value and then selects an action correspondingly. In order to compress the huge action space in RLcard, we split the DQN for one integral action space into two DQNs for main-group and kicker-card action spaces respectively. In Fig. 4, the process of Two-steps Q-network is as follows: given a state, the main-group Q-network works and produces a Q-value, and according to this Q-value, action a1, which will decide both the main-group hands to play and whether the kicker-card Q-network will be activated, will be selected using epsilon-greedy policy. Then, if the kicker-card Q-network has been activated, its output, *i.e.*, an action denoted as "a2", decides the kicker-cards to play. Finally, the produced main-group hands and kicker-cards form the integrated cards to be played. To summarize, the main-group Q-network and the kicker-card Q-network make joint effort to help the agent take a complete action under the current state to play. To illustrate the process of two-steps Q-network, we take the final action 3334 as an example. In two-steps Q-network, the main-group Q-network is responsible of choosing the action of 333* and activates the kicker-card Q-network. Then the kicker-card Q-network takes charging of choosing the action of ***4. After main-group Q-network and kicker-card Q-network produce actions sequentially, a final action-3334 is generated and played out by two-steps Q-network.

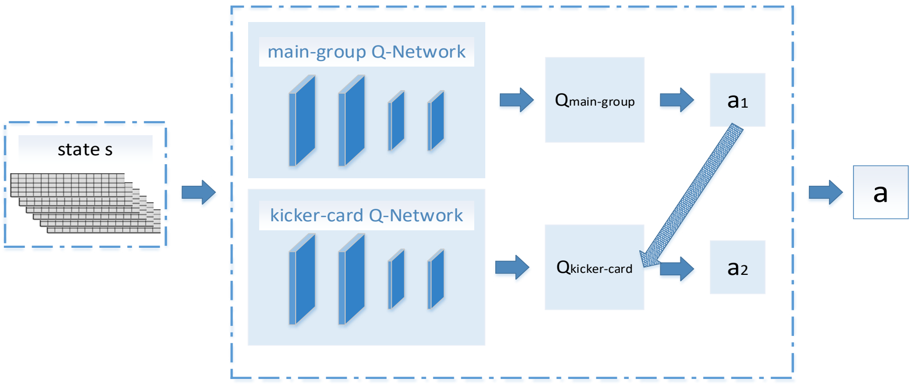


Fig. 4. The structure of two-steps Q-network

Main-group Q-network. The policies for the three roles (*i.e.*, the landlord, the down-peasant and the up-peasant) in Dou Di Zhu are different, which makes their reward functions be different and thus, three different main-group Q-networks are trained for them respectively. Some training data <*s*, *a*, *r*, *s*'>are collected from high-level players and tournaments and used to gain reward functions through deep apprenticeship learning (DAL) [28]. DAL has two main parts: deep apprenticeship Q-network (DAQN) and deep apprenticeship reward network (DARN) and they are train sequentially. In DAQN, <*s*, *a*> is taken as input and the softmax prediction of the possible actions under the state is the output. The loss function used in DAQN is shown in Eq. (5).

$$J_Q(w_q) = \sum_a \left[q_{w_q}(s,a) - \hat{q}(s,a) \right]^2 , \tag{5}$$

where w_q is the learned weights, $q_{w_q}(s,a)$ denotes the softmax output of the DAQN and $\hat{q}(s,a)$ represents the actual action taken by the expert which is abstracted into a one-hot array. In $\hat{q}(s,a)$, the actual action is assigned with the value of 1 and 0 the otherwise. DARN is used to abstract the reward function from DAQN. The loss function used in DARN is shown in Eq. (6).

$$J_R(w_r) = \left\| r_{w_r}(s,a) - \hat{r}(s,a) \right\|_2 . \tag{6}$$

Where w_r is the learned weights, $r_{w_r}(s,a)$ is the output of the DARN and $\hat{r}(s,a)$ is the current value of (s,a) which is calculated from the DAQN.

After DARN is well train, the reward of the action under the state is produced, as is shown in Eq. (7).

$$r(s,a) = w_r \cdot f(s,a), \tag{7}$$

where w_r is learnt from DARN and varies for the three different roles. Consequently, the initial rewards given by the environment of Dou Di Zhu are replaced by the rewards produced by the generated reward functions above. Then, the consequent samples are updated and collected into replay memory to train main-group Q-network.

It is worth noting that the existing implementations of DAL treat DRQN and DARN as two completely separate parts. That is to say, the DARN network starts with random weights and biases to learn from the DAQN network which will cost a lot of time, thus transfer learning is introduced to solve the problem. We copy the first several layers in DAQN which are much more related to the analysis of information of state to construct DARN, in order to ensure that DARN's main attention is on calculating the state-action values. Transfer learning and fine-tuning are very useful in reducing the expensive cost of data analysis and in alleviating the overfitting problem [29].

Kicker-card Q-Network. Reward shaping is used to build the reward function for the kicker-card Q-network. Besides, in order to train kicker-card Q-network effectively, we use prioritized experience replay to make better utilization of valuable transitions.

The Definition of Reward Function. In real Dou Di Zhu games, human-being players always prefer playing the kicker-cards which have slighter influence on their following actions, and thus the reward function of a kicker-cards combination is accordingly defined as:

$$reward_{rule} = k \cdot e^{-(a+b)}, \tag{8}$$

where a is the count of all the possible legal combinations that could be formed by the kicker-cards and other cards, and b is the sum of all the ranks of the kicker-cards. Based on this, it is reasonable to assume that $a + b$ correlates with the importance of a kicker-cards combination and thus negatively correlates with the reward of the combination as well, as defined in Eq. (8).

Prioritized Experience Replay for Kicker-card Q-Network. The reward function defined for kicker-card Q-network is related with the specific cards, thus the reward of pass (that is, no cards) could not be obtained from the reward function, and the rewards in kicker-card Q-network will still be parse consequently. Against this, we use PER to make a difference among the transitions and give the more valuable transitions higher priorities. In this work, an array of commonly actions taken by human-beings is defined, and the transitions (samples) resulted from these actions will be assigned with higher priorities based on TD-ERROR. The priority is defined as:

$$p_i^{'} = p_i + \eta, \tag{9}$$

where η is a Boolean variable to represent whether the action that results to transition i is in the array of common kicker-cards introduced above.

4 Experiments

Many recent researches have focused on introducing AI technics into poker games. Counterfactual regret minimization (CFR) and its variants were used in poker games like Texas Hold'em Poker [30-33]. Libratus computes a blueprint for the overall strategy and fixes potential weaknesses identified by the opponents in the blueprint strategy [6]. DeepStack adopts a neural network to approximate the tabular CFR and recursive reasoning [7]. AlphaHoldem is proposed with an end-to-end self-play reinforcement learning framework [8]. However, Methods based on CFR heavily depend on complete traversals of the game trees which are difficult to be obtained in Dou Di Zhu due to the huge state and action spaces. Recently, researches on Dou Di Zhu have seen great success. RLcard, a simulation platform, implements DQN and NFSP [34] algorithm for Dou Di Zhu. Unfortunately, the performance of DQN on RLcard is quite unsatisfactory. Combinational Q-learning (CQL) addressed the challenging issue of huge action space through employing a two-stage combinational Q-learning to decompose cards. However, the time used to decompose the cards is up to the amount of the cards, and the effect of decomposition quite relies on human heuristics [14]. DeltaDou makes inferences of the unseen information using Baysesian methods, and applies the obtained inference results in Fictitious Play MCTS to form an AlphaZero-like framework [15]. DouZero is a quite satisfactory for Dou Di Zhu, which enhances classic MC methods with DNNs. However, it treats the kicker-cards and main-groups equally, which means that DouZero ignores the huge action space to a certain extent. Besides, MC methods are based on substantial samples and complete episodes of games as well. Our work adopts temporal difference (TD) which is more applicable to the situations with huge state and action spaces.

In this section, we firstly introduce the criteria, baseline and settings of our experiments, then we compare and analyze the performances of Two-steps Q-network and DQN in RLcard in terms of winning probability and stability.

4.1 Experimental Criteria and Baseline

Winning rate: the proportion of the winning rounds in the total rounds of game. Then, Performance stability: evaluated in term of the variance of the last five winning rates obtained in the last five points in time respectively, and it is notable that we still compute the variance when all the available winning rates are less than five. The experiments are carried out on RLcard which is a toolkit for RL research in card games. DQN is used as the benchmark of our experiment and for which several rules are given in advance in RLcard: only the final actions leading to victory are given the reward of 1, and the rewards of all the other actions are considered to be zero.

4.2 Experimental Settings

Both two-steps Q-network and DQN are set on the landlord side, the down-peasant side and the up-peasant side respectively and compete with random agents. All the experiments are carried out on a single server of Intel Core i9-9900K CPU 3.60 GHz and 32.0 GB memory. The hyper-parameters shared in two-steps Q-network and DQN, as shown in Table 1.

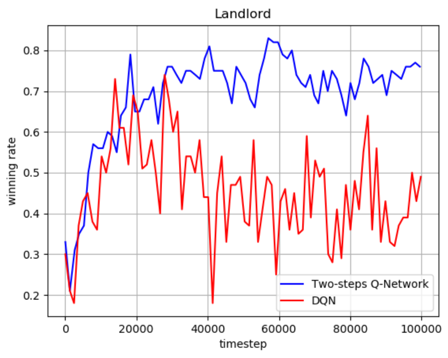
Table 1. The hyper-parameter settings

Hyper-parameter	Value
memory-size	20000
discount-factor	0.99
learning-rate	0.00005
batch-size	32
the size structure of fully-connected-layers	64-64-32-32

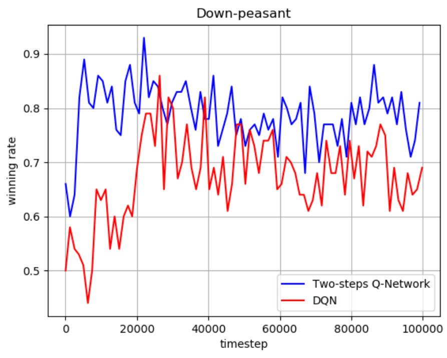
4.3 Results Analysis

In generally, two-steps Q-network surpasses DQN, as shown in Fig. 5. Specifically, from Fig. 5(a) to Fig. 5(c), the winning rate of two-steps Q-network varies between 70% and 80% at the side of the landlord which is almost 30% higher than the counterpart in DQN. The winning rate of two-steps Q-network keeps around 80% at the both sides of the down-peasant and the up-peasant. What's more, in order to evaluate the designed reward functions in two-steps Q-network, we test the performances of variants of two-steps Q-network through removing the reward functions one by one. Figure 5(d) shows the comparison between the performance of DQN and the variant of two-steps Q-network without the reward functions for both main-group Q-network and kicker-card Q-network, and it can be seen that the performance of the variant of two-steps Q-network is inferior to DQN at the initial stage of training and gets roughly the same after a period of training. Figure 5(e) demonstrates the comparison results between two-steps Q-network and its two variants without the reward function in the main-group Q-network and that in kicker-card Q-network respectively. From the experimental results show that the two-step Q-network outperforms both of the two variants, which can prove the efficiency of the reward functions in both main-group Q-network and kicker-card Q-network.

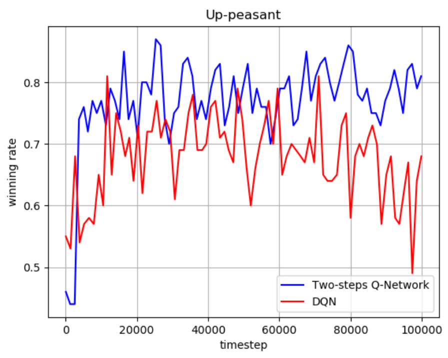
Two-steps Q-network outperforms DQN in stability in terms of the variance of winning rate as well, as shown in Fig. 6. In specific, from Fig. 6(a) to Fig. 6(c), the variances of the winning rates of two-steps Q-network for all the three roles (*i.e.*, the landlord, the down-peasant, and the up-peasant) decrease after a huge increase while those of DQN keep comparatively unstable and higher. The great variance in DQN indicate that DQN is not trained well and cannot steadily take wise actions even after being trained for a long period. The possible reasons of the undesirable performance of DQN include the sparse rewards and the large action space. In Dou Di Zhu, there is only one non-zero reward (if any) that appears at the end of each round of game, and this makes the training of DQN be slow and unstable. What's more, the huge action space of Dou Di Zhu makes the training of DQN unsatisfactory. In comparison, Two-steps Q-network solves these two challenges to some degree by constructing proper reward functions and compressing the huge action space respectively.



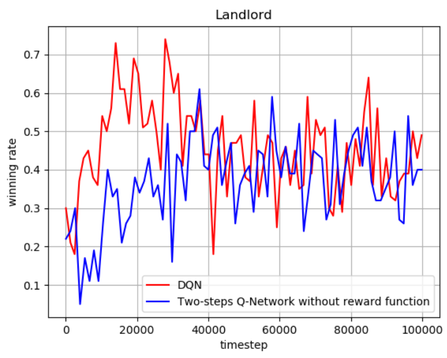
(a) Winning rate of the landlord



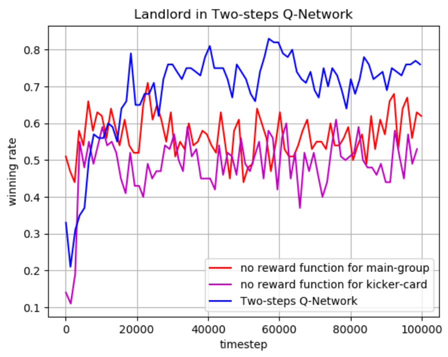
(b) Winning rate of the down-peasant



(c) Winning rate of the up-peasant

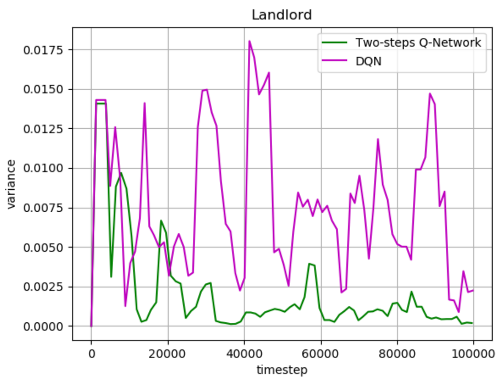


(d) Two-steps Q-network without defined reward function VS DQN

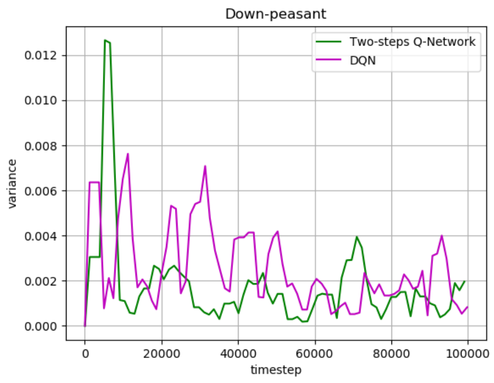


(e) Two-steps Q-network VS no reward functions

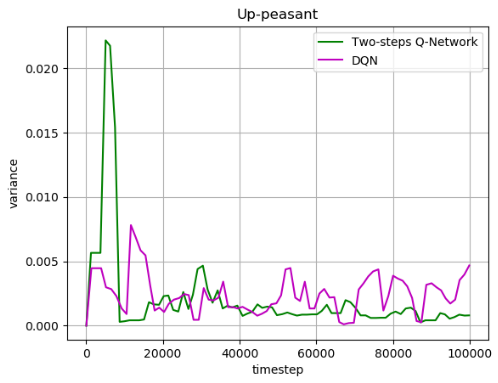
Fig. 5. Winning rate of different situations



(a) Variance of the landlord



(b) Variance of the down-peasant



(c) Variance of the up-peasant

Fig. 6. Variance of different situations

5 Conclusion

This work tries to overcome the problems of introducing DRL methods into Dou Di Zhu. First of all, this work decomposes initial Q-network which has a huge action space into kicker-cards Q-network and main-group Q-network, motivated from the fact that there are 20,000 legal combinations in action space and the huge action space will do harm to the performance of DRL methods. Second, in order to solve the problem of sparse and delayed rewards, this work utilizes different approaches in light of the main-group Q-network and the kicker-card Q-network. For the main-group Q-network, instead of building one general reward function for all agents in different sides, we first classify the demonstrations by roles and then gain three different reward functions with respect to the roles through DAL. In our work, reward shaping, and prioritized experience replay are used to solve the problem of the sparse reward in kicker-card Q-network.

For future work, we will conduct further researches from the following aspects. First, Dou Di Zhu is a typical research platform for multi-agent RL which includes cooperation and competition, we will introduce cooperation into the two peasants. Second, we are prepared to take advantage of other neural network architectures in Dou Di Zhu, such as convolutional neural networks, long short-term memory, and ResNet. Third, since rewards are of vital significance for DRL, we plan to explore other approaches to solve the problem of sparse reward and improve the winning rate.

References

[1] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529(7587)(2016) 484-489.

[2] DeepMind, Alphago. <https://www.deepmind.com/research/case-studies/alphago-the-story-so-far#the_matches>, 2017 (accessed 05.01.22).

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, Nature 550(7676)(2017) 354-359.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, Mastering chess and shogi by self-play with a general reinforcement learning algorithm. <https://arxiv.org/abs/1712.01815>, 2017 (accessed 20.10.21).

[5] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering atari, go, chess and shogi by planning with a learned model, Nature 588(7839)(2020) 604-609.

[6] N. Brown, T. Sandholm, Superhuman AI for heads-up no-limit poker: Libratus beats top professionals, Science 359(6374)(2018) 418-424.

[7] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, Deepstack: Expert-level artificial intelligence in heads-up no-limit poker, Science 356(6337)(2017) 508-513.

[8] E. Zhao, R. Yan, J. Li, K. Li, J. Xing, AlphaHoldem: High-Performance Artificial Intelligence for Heads-Up No-Limit Poker via End-to-End Reinforcement Learning, in: Proc. 2022 the AAAI Conference on Artificial Intelligence, 2022.

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

[10] J.H. Chang, H.H. Chiang, H.X. Zhong, Y.K. Chou, Travel Package Recommendation Based on Reinforcement Learning and Trip Guaranteed Prediction, Journal of Internet Technology 22(6)(2021) 1359-1373.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning. <https://arxiv.org/abs/1312.5602>, 2013 (accessed 03.10.20).

[12] T. Zahavy, M. Haroush, N. Merlis, D.J. Mankowitz, S. Mannor, Learn what not to learn: Action elimination with deep reinforcement learning. <https://arxiv.org/abs/1809.02121>, 2018 (accessed 20.11.21).

[13] S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in: Proc. 1993 the Fourth Connectionist Models Summer School, 1993.

[14] Y. You, L. Li, B. Guo, W. Wang, C. Lu, Combinational Q-Learning for Dou Di Zhu. <https://arxiv.org/abs/1901.08925>, 2019 (accessed 03.05.21).

[15] Q. Jiang, K. Li, B. Du, H. Chen, H. Fang, DeltaDou: Expert-level Doudizhu AI through Self-play, in: Proc. 2019 the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2019.

[16] Y.M. Shi, Z. Zhang, Research on Path Planning Strategy of Rescue Robot Based on Reinforcement Learning, Journal of Computers 33(3)(2022) 187-194.

[17] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, J. Liu, DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning. <https://arxiv.org/abs/2106.06135>, 2021 (accessed 14.06.21).

[18] A.Y. Ng, S.J. Russell, Algorithms for inverse reinforcement learning, in: Proc. 2000 International Conference on Machine Learning, 2000.
[19] C.J.C.H. Watkins, P. Dayan, Q-learning, Machine learning 8(3-4)(1992) 279-292.
[20] J. Zhang, C. Zhang, W.C. Chien, Overview of deep reinforcement learning improvements and applications, Journal of Internet Technology 22(2)(2021) 239-255.
[21] B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and autonomous systems 57(5)(2009) 469-483.
[22] S. Schaal, Is imitation learning the route to humanoid robots?, Trends in cognitive sciences 3(6)(1999) 233-242.
[23] P. Abbeel, A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proc. 2004 the twenty-first International Conference on Machine learning, 2004.
[24] L.J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching, Machine learning 8(3-4)(1992) 293-321.
[25] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, <https://arxiv.org/abs/1511.05952>, 2015 (accessed 03.11.20).
[26] Y. Hou, L. Liu, Q. Wei, X. Xu, C. Chen, A novel DDPG method with prioritized experience replay, in: Proc. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017.
[27] D. Zha, K.H. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, X. Hu, Rlcard: A toolkit for reinforcement learning in card games. < https://arxiv.org/abs/1910.04376>, 2019 (accessed 01.05.21).
[28] M. Bogdanovic, D. Markovikj, M. Denil, N. Freitas, Deep apprenticeship learning for playing video games, in: Proc. Workshops at the AAAI Conference on Artificial Intelligence, 2015.
[29] D.Y. Huang, W. L. Lin, Y.Y. Wang, Approaches of Transfer Learning and Fine-Tuning on the Effects of Performance of Vehicle Classification, Journal of Computers 31(6)(2020) 24-37.
[30] M. Zinkevich, M. Johanson, M. Bowling, C. Piccione, Regret minimization in games with incomplete information, in: Proc. Advances in neural information processing systems 20 (NIPS 2007), 2007.
[31] N. Brown, T. Sandholm, Solving imperfect-information games via discounted regret minimization, in: Proc. 2019 the AAAI Conference on Artificial Intelligence, 2019.
[32] M. Lanctot, E. Lockhart, J.B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, J. Ryan-Davis, OpenSpiel: A framework for reinforcement learning in games. <https://arxiv.org/abs/1908.09453>, 2019 (accessed 03.11.21).
[33] K. Li, H. Xu, M. Zhang, E. Zhao, Z. Wu, J. Xing, K. Huang, OpenHoldem: An Open Toolkit for Large-Scale Imperfect-Information Game Research. <https://arxiv.org/abs/2012.06168v1>, 2020 (accessed 20.10.21).
[34] J. Heinrich, D. Silver, Deep reinforcement learning from self-play in imperfect-information games. <https://arxiv.org/abs/1603.01121>, 2016 (accessed 12.11.21).