Neural Processing Letters (2023) 55:9467–9482
https://doi.org/10.1007/s11063-023-11209-0

Application of DQN-IRL Framework in Doudizhu's Sparse Reward

Yan Kong[1] · **Hongyuan Shi**[1] · **Xiaocong Wu**[1] · **Yefeng Rui**[1]

Accepted: 24 February 2023 / Published online: 20 March 2023

Abstract
When applying Artificial Intelligence into the traditional Chinese poker game Doudizhu, it is faced with many challenging issues resulted from the characteristics of Doudizhu. One of these challenging issues is the sparse reward, due to the truth that a valid feedback could be obtained only at the end of a round of the game. Against this, in this paper, a deep neural framework, DQN-IRL, is proposed to address the challenging issue of sparse reward in Doudizhu. The experimental results proves the efficiency of DQN-IRL (Inverse Reinforcement Learning) in terms of winning rate.

Keywords Doudizhu · Reinforcement learning · Sparse reward · Inverse reinforcement learning

1 Introduction

Nowadays, AI(Artificial Intelligence) has been widely used and seen great success in game fields, such as Go, Atari and Starcraft, and it even surpassed the best human players. In the field of Go, there are some well known AI-based games, such as AlphaGo [1], AlphaGo Zero [2] and AlphaZero [3]. The AI research environment ALE (Arcade Learning Environment) [4] is developed for famous Atari game series and SC2LE(StarCraft II Learning Environment) [5] is developed for StarCraft II which is jointly released by DeepMind and Storm Snow. These three AI are built based on the AI algorithm MCTS(Monte Carlo Search Tree) [6] and deep neural network. The famous Texas Hold 'em Poker AI is Pluribus [7], jointly developed by The artificial intelligence team of Facebook and the computer Science department of Carnegie Mellon University. Two years ago they developed a poker system called Libratus [8], which used to be a world champion in Texas Hold 'em singles tournaments. In the field of Doudizhu, many powerful AI have indeed emerged, such as DeltaDou [9], CQN (Combinational Q-Learning Network) [10] and DouZero [11], all of which have reached the level of human players. Currently the strongest AI is DouZero developed by Kuaishou. These three kinds of AI adopt different AI algorithms, including MCTS, deck decomposition

✉ Yan Kong
kongyan4282@163.com

1 Nanjing University of Information Science and Technology, Nanjing, China

combination and DMC (Deep Monte Carlo). DeltaDou proposed FPMCTS (Fictitious Play MCTS) based on MCTS algorithm. In the expansion stage of MCTS, this method extends the nodes by two levels and carries out Monte Carlo simulation. With the increase of simulation times, the probability of action space is more accurate, but it consumes more computing resources. The CQN algorithm mainly completes the game by decomposing and selecting the card type by dividing the whole strategy process into two steps: DPN (Decomposition Proposal Network) and MPN (Move Proposal Network), and performing the card split task and the action selection task respectively. The effect of CQN is good, but it is time consuming and resource expensive. What's more, because there are too many card types in Doudizhu, CQN is difficult to converge. DouZero is one of the state-of-the-art frameworks in the field of Doudizhu, in which Monte Carlo method with neural network is adopted. The structure is simple and effective, and the training time and computing resources are less than both DeltaDou and CQN. However, the problem is that in the face of some very simple situations, DouZero only relies on the rules of large numbers to make a completely wrong choice and does not have enough flexibility. The existing researches have indeed achieved quite good results in Doudizhu, but they are not perfect in fault detection, the influence of interference, modeling error, various uncertainties in the real systems and so on. Because of Doudizhu can be regarded as a Markov jump system to some extent, and different styles of playing cards are regarded as different modes. When the styles of playing cards change suddenly, the above methods do not have the ability of fault detection [12] and anti-interference [13], and will still rely on MCTS and the rules of large numbers for playing cards, which is not correct in many cases. Moreover, Doudizhu is a multiplayer game [14], and we need to learn some new methods to solve this problem in the future. On the other hand, these methods do not propose proper solutions to the special sparse reward problem in Doudizhu.

Sparse reward is a challenging issue in the field of reinforcement learning, and many researchers have proposed different solutions, such as reward shaping [15], curriculum learning [16] and HRL (Hierarchical Reinforcement Learning) [17]. When it comes to solving some special problems, reward shaping is indeed a method worth trying. But there is a prerequisite that we need to know the domain knowledge which is quite difficult because, in many cases, we cannot find human beings who are proficient in a certain field to cooperate, as a result, the limitation of this scheme is relatively high. Curriculum learning, a more popular method in recent years, mainly divides the training into many stages, from simple training, step by step. This method can accelerate the training of machine learning model, make the model get better generalization, and reach the local optimal value faster and better. The key of Curriculum Learning is the settings of curriculum in which different samples need to be artificially set for different problems. Because Doudizhu is a random game, and, in different episodes, the difficulty of the same sample is different. It is clear that the most important curriculum design in curriculum learning is quite difficult when it is adopted into Doudizhu. The goal of hierarchical reinforcement learning is to divide a task into multiple sub-tasks, which can greatly improve the utilization efficiency of samples, so as to alleviate the problem of sparse reward in the total task. HRL is mainly divided into two kinds, one is goal-based [18], the other is multi-level control [19] all of which require the decomposition of an overall task, but some problems cannot be broken down at all, e.g., fighting landlords which is a problem that cannot gain the reward when a certain action is taken even if it is decomposed, and this phenomenon is regarded as hard sparse reward. To conclude, it is difficult to take mild measures to solve the problem of sparse reward. Moreover, most of these methods need complex artificial design for reward before training, while reward function itself is an objective existence, taking too much artificial design is a restriction that affects the final scientific nature of the algorithm.

Therefore, we expect to adopt IRL (Inverse Reinforcement Learning) [20] to make the agent learn a scientific and effective reward function from the trajectories of experts, which not only avoids the complex and tedious manual design, but also gets a relatively more scientific and accurate reward function. What IRL learns is more like the overall policy when facing a certain problem than the piecemeal stage policy, which enables IRL better deal with the parse reward problem, instead of relying the algorithm on the rules of large numbers.

Against this background, this paper builds the DQN-IRL learning framework and applys it in Doudizhu, so as to analyze the effect of inverse reinforcement learning in solving the hard sparse reward problem. DQN-IRL framework is mainly built according to inverse reinforcement learning and DQN (deep Q-Network) [21].

The contributions of this paper include the following two points:

1. Building a reasonable and effective DQN-IRL framework to solve the hard sparse reward problem in Doudizhu.
2. Through the verification of experimental results and literature analysis, this paper provides a new solution to the hard sparse reward problem with fuzziness and randomness (e.g., Doudizhu).

2 Background

2.1 Doudizhu

Doudizhu is a three-player (i.e., a landlord and two peasants) game in which the three players bid to decide who will be the landlord. At the beginning, the landlord has 20 cards and each peasant has 17 ones, and there are a set of various legal cards combinations to play for each player at each step. All the legal cards combination can be played are shown in Table 1.

2.2 IRL

IRL has been used in automatic driving [22]. As there are many situations encountered in the process of driving, it is impossible to input all possible situations into the trajectory. And the randomness and uncertainty in the process of car driving, the reward function is also difficult to define. Therefore, agents need to learn on their own what to do when they meet situations other than expert data, which has something in common with Doudizhu.

Inverse reinforcement learning [20] can obtain a reward function through expert data, and take the learned reward function as the reward function in DQN [21] algorithm to help agents update network parameters effectively in the learning process and improve the performance of DQN algorithm. By comparing the expected reward obtained by the current reward function with the expected reward of the expert data, the new reward function can be obtained again, and the reward function that is closest to the expert behavior can be obtained by successive iterations.

Inverse reinforcement learning is to solve the problem that the reward function is difficult to define in some scenes, and to restore or approximate the reward function by observing the behavior trajectory of experts. The framework of inverse reinforcement learning is shown in Fig. 1.

Table 1 The examples of action in Doudizhu

Type	Definition	Examples
Solo	Just one card	"3" or "4"
Pair	Two of the same cards	"33" or "44"
Trio	Three of the same cards	"333" or "444"
Trio with single	Three of the same cards with a different card	"3334" or "9995"
Trio with pair	Three of the same cards with a pair	"33,344" or "9995"
Chain of single	Five or more consecutive cards (except 2, Black joker and Red joker)	"34,567" or "89TJQ"
Chain of pair	Three or more consecutive pairs (except "2", Black joker and Red joker)	"334,455" or "445,566"
Chain of trio	Two or more consecutive trio (except "2", Black joker and Red joker)	"333,444" or "555,666"
Plane with sole	Two or more consecutive "Trio with Single"	"33,344,456" or "77,788,849"
Plane with pair	Two or more consecutive "Trio with pair"	"3,334,445,566" or "7,778,884,499"
Quad with solo	Four of the same cards with a card	"33,334" or "55,556"
Quad with pair	Four of the same cards with a pair	"333,344" or "555,566",
Bomb	Four of the same cards	"3333" or "5555"
Rocket	Black joker and Red joker	
PASS	Give up in this round	

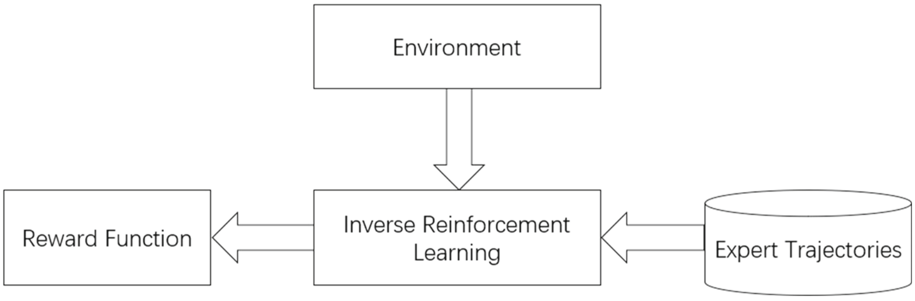

Fig. 1 IRL Frame work

2.3 DQN (Deep Q Network).

DQN algorithm is proposed by DeepMind team. As a classical algorithm in the field of DRL (Deep Reinforcement Learning), it is an outstanding and representative algorithm that combines deep learning and reinforcement learning. In the field of reinforcement learning, high-dimensional input data have always been a challenging issue. Before DQN, most reinforcement learning cases were completed by manually designing features, so that the performance of the algorithm depends much on the quality of manually designed features, which was undoubtedly troublesome. DQN makes it possible for reinforcement learning to extract features from raw data by applying neural network to reinforcement learning. The neural network used by DQN algorithm is trained by the variant Q-learning algorithm, and

SGD (Stochastic Gradient Descent) is used to update the weights, and the experience replay mechanism is used to eliminate the correlation between data through randomly sampling past transitions.

In Q-learning, $Q(s, a)$ (s represents state and a represents action) is used to represent the Q value, but in DQN, $Q(s, a; \theta)$ is used to represent the Q value where θ is the parameter in the neural network. Because it is difficult to realize Q table in Q-learning algorithm when input data is high-dimensional, DQN changes the update problem of Q table into a function fitting problem. By updating parameter θ, Q function approaches the optimal Q value:

$$Q(s, a; \theta) \approx Q'(s, a) \tag{1}$$

Then the Loss Function of the updated network parameter θ is determined based on the update formula of Q-learning:

$$L(\theta) = E\left[(TargetQ - Q(s, a; \theta))^2\right] \tag{2}$$

$$TargetQ = r + \gamma max Q\left(s', a'; \theta\right) \tag{3}$$

where r is the reward and γ is the discount factor.

2.4 DQN-IRL Framework

In this section, we firstly introduce the encoding way used in this work, followed by specifically introducing our proposed framework DNQ-IRL.

2.5 Encoding of State and Action

In our work, a state is defined as all information observed by a player agent within a certain period of time, including the current cards, the cards of the other two players, the recent three actions and all cards that have been played. The state S of the player is defined as a matrix set consisting of m matrixes, and the size of each of these matrixes is n*k, where n*k represents the account of the kth type of card is n. For example, the given cards set is 233444JJK, as Fig. 2 shows, and the corresponding encoding is shown in Table 2. In our work, m, n, and k are 6, 5, and 15 respectively. In specific, the 6 matrixes respectively are the current cards (denoted by 1 matrix), the cards of the other two players (denoted by 1 matrix), the recent three actions (denoted by 3 matrixes) and all cards that have been played (denoted by 1 matrix). An action is defined as a 5*15 matrix, which means the same with that in a state (Table 2).

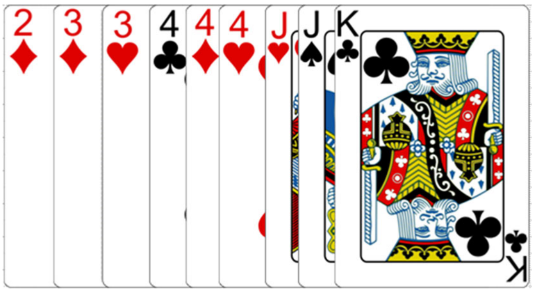
Fig. 2 An example of cards, the encoding of it is shown in Table 2

Table 2 The encoding of the card type and number

Count/card	2	3	4	J	K	Other cards
0	0	0	0	0	0	1
1	1	0	0	0	1	0
2	0	1	0	1	0	0
3	0	0	1	0	0	0
4	0	0	0	0	0	0

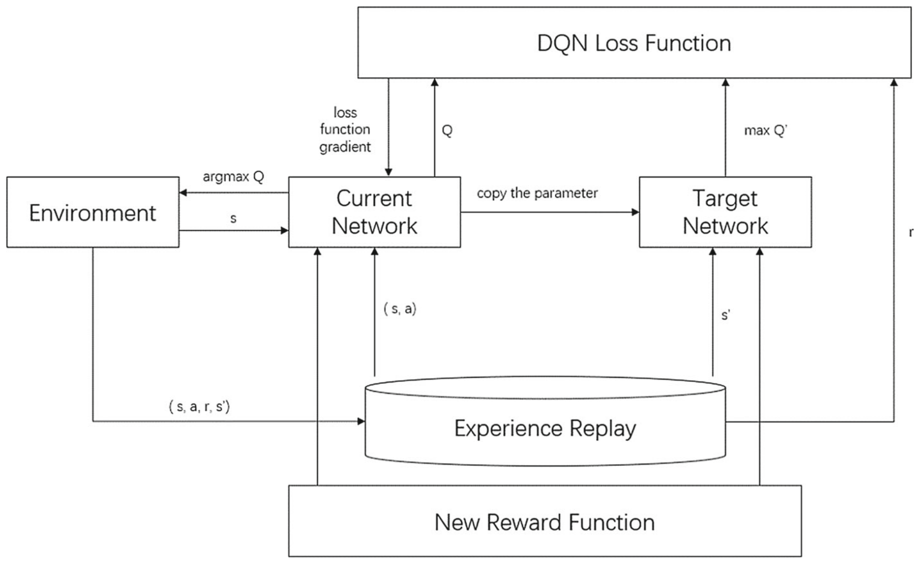

Fig. 3 DQN-IRL Frame work

2.6 DQN-IRL Framework

Since rewards in Doudizhu are sparse and reward functions are difficult to define, direct application of DQN algorithm in Doudizhu does not achieve good results. We hope agents can learn appropriate reward functions by themselves, and apprenticeship learning proposed by P Abbeel and AY Ng [23] is adopted in this work to learn reward functions. Apprenticeship learning is a kind of maximum reinforcement learning. Apprenticeship learning consists of two principal steps, the first step is to calculate the parameters of the reward function by using the method of maximum margin in the optimal policy obtained by iteration, and the second one is to conduct positive reinforcement learning according to the reward function to find the optimal policy, and then repeat the first step. The reward function $R(s)$ in this paper is based on apprenticeship learning, TD (Temporal Difference) is employed to update Q values (Eqs. (2) and (3)), and r in formula (3) is derived from $R(s)$. The framework of DQN-IRL is shown in Fig. 3 above:

Reward Function in Fig. 3 is the Reward Function learned in Fig. 1, which is put into two value networks (the Current Network and the Target Network) in DQN to help agents calculate Q values. DQN Loss Function is obtained by Eq. (2) and Eq. (3).

When applying DQN-IRL framework to Doudizhu, expert trajectory $\{\tau_1, \tau_2, \tau_3, \ldots \tau_n\}$ is provided to learn a reward function, which is obtained through apprenticeship learning:

$$R(S) = w \cdot \emptyset(S) \tag{4}$$

where S is defined as a matrix set consisting of m matrixes each of which the size is n*k, in which n*k represents the account of the kth type of card is n. And w is a randomly initialized parameter set. $\varnothing(s)$ is the basis function, which can be a polynomial basis or a Fourier basis, and a polynomial basis is adopted in this paper.

The framework based on DQN-IRL needs to obtain the values of w in Eq. (4) first. According to the definition of value function, the value function of the selecting random policy π is:

$$E_{s_0 \sim D}\left[V^{\pi}(s_0)\right] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right] = E\left[\sum_{t=0}^{\infty} \gamma^t w \cdot \emptyset(s_t)|\pi\right] = w \cdot E\left[\sum_{t=0}^{\infty} \emptyset(s_t)|\pi\right] \tag{5}$$

where D is the initial-state distribution, s_0 is the start state, and t is timestep.

According to the change of policy π, feature expectation will also change, and the feature expectation is defined as:

$$\mu(\pi) = E\left[\sum_{t=0}^{\infty} \gamma^t \emptyset(s_t)|\pi\right] \tag{6}$$

The value function can be rewritten as:

$$E_{s_0 \sim D}\left[V^{\pi}(s_0)\right] = w \cdot \mu(\pi) \tag{7}$$

At this point, given n expert trajectories, the feature expectation of expert policy can be estimated according to:

$$\hat{\mu}_E = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{\infty} \gamma^t \emptyset\left(s_t^{(i)}\right) \tag{8}$$

where $s_t^{(i)}$ is the collecting expert trajectory $\{s_0, s_1, s_2, \ldots\}_{i=1}^{n}$.

Random policy $\widetilde{\pi}$ and expert policy satisfy the following inequality in which ϵ is a hyper-parameter:

$$\|\mu(\tilde{\pi}) - \mu_E\|_2 \leq \in \tag{9}$$

After finding the parameter w under the current optimal policy, figure out $r(\varnothing)$ from formula $r(\varnothing) = w^T * \mu(\pi)$, according to the framework of DQN-IRL, we have:

$$TargetQ = r(\emptyset) + \gamma max Q\left(s', a'; \theta\right) \tag{10}$$

According to the reward function, the optimal policy is continued to be worked out until the feature expectation of the expert trajectory is approached. At this point, the reward function closest to the expert policy can be obtained.

3 Experiment

In this paper, RLCard toolkit [24] is used as the experimental platform, which is a solution and opened source platform provided by Data Science Laboratory of Texas A&M University to help related researchers implement their ideas. RLCard is a platform for reinforcement learning based on card game design, which integrates several of the most popular card games in China and the West (including Doudizhu, Mahjong, blackjack, Texas Hold 'em, UNO, etc.). RLCard also implements some reinforcement learning algorithms, which is committed to providing an easy-to-use, unified development and testing environment for reinforcement learning, so that people can easily and conveniently train and test their ideas. In the experiment, we choose DQN and NFSP (Neural Fictitious Self-Play) [25] as the benchmarks, and the DQN algorithm has been explained in the previous DQN section. NFSP is a model of learning approximate Nash equilibrium in imperfect information game, which combines fictitious play with deep learning. In NFSP, players respond best to other players' average policy through DQN in each iteration and update their average policy through supervised learning.

3.1 Experiment Settings

About 1000 samples of expert data were collected for this experiment, all of which came from randomly selected professional players, so as to avoid unnecessary impact on the experimental results. In the experiment, we also tried to use different neural network parameters for training, and found that the final training results were not very different. In order to avoid the influence of manual setting parameters, we finally chose randomized parameters for training. NVDIA 1650Ti GPU and 8G operating memory were used for the training. No matter in the game world or in the real world, effectiveness and robustness are the two most important indicators to evaluate a good algorithm, which can be depicted by winning rate and variance of winning rates simply and effectively, therefore, winning rate and the variance are chosen as the evaluation criteria of the performance of this framework. The experimental process is carried out according to the framework of DQN-IRL. In specific, IRL is used to find a reward function through 1000 expert data (i.e., 1000 expert trajectories) we collected, and the reward function is expressed as $R^*(s) = w^* \times \varnothing(s)$, $(w^* \in R^k)$, where $\varnothing(s)$ represents feature vector, w^* denotes the weights, and $R^*(s)$ is the reward function. IRL is used to constantly update the reward function according to the difference between the expectations obtained from the old reward function and the expectation of experts respectively. Finally, the reward function closest to expert behavior is obtained.

3.2 Experiment Results and Analysis

DQN-IRL is evaluated from the positions of landlord and peasant respectively, through being compared with DQN and NFSP. The winning rate and the corresponding variance are shown in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 15.

According to Figs. 4 and 5, it can be seen that the winning rate of DQN-IRL in the position of the landlord is mostly higher than that of DQN. As to the stability comparison, the variance of winning rate in DQN and DQN-IRL is close. Although the variance of DQN-IRL is larger in the early stage, it becomes slightly lower than that of DQN in the later stage, as shown in Fig. 5.

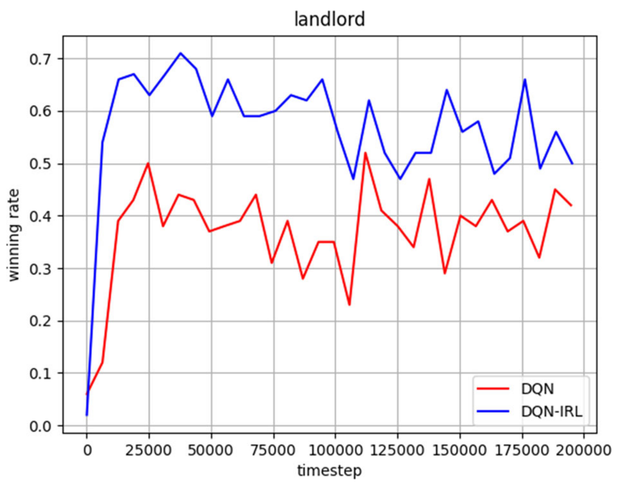

Fig. 4 DQN-IRL and DQN as landlord

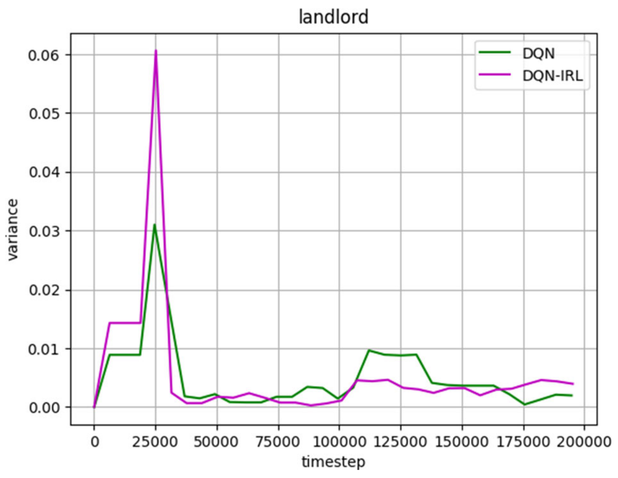

Fig. 5 The variance of DQN-IRL and DQN as landlord

According to Figs. 6 and 7, it can be seen that DQN performs better than NFSP in terms of winning rate, and the difference of their variance is quite small, which demonstrates that DQN performs better than NFSP while they both perform stably. Therefore, it can be seen that it is practical and effective to use inverse reinforcement learning to help agents learn how to play cards better.

From Fig. 9, compared with DQN, the variance of DQN-IRL is smaller, which demonstrates that DQN-IRL has a more stable performance. According to Figs. 10 and 11, the performance of DQN-IRL is better than NFSP on both winning rate and variance.

However, we found that when being in different positions, the performance of the DQN-IRL agent varies greatly, and the performance of agent in the role of up peasant makes this more obvious.

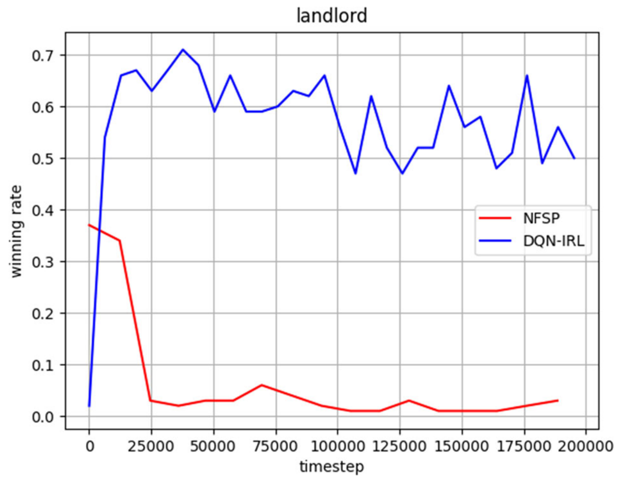

Fig. 6 DQN-IRL and NFSP as landlord

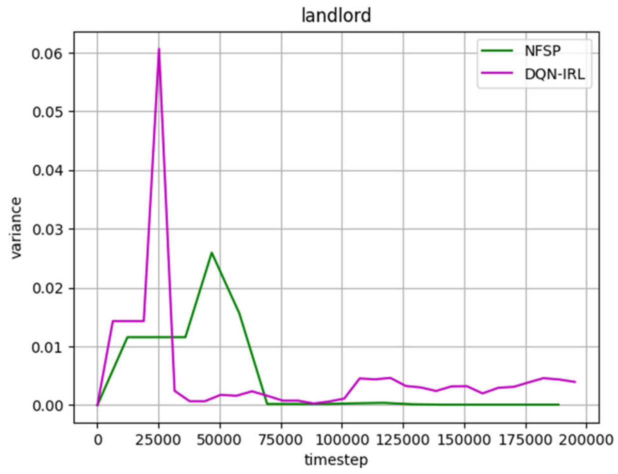

Fig. 7 The variance of DQN-IRL and NFSP as landlord

It can be seen from Figs. 12 and 13 that compared with DQN, the performance of DQN-IRL is not significantly improved in the role of the up peasant. In Doudizhu, the policy of peasant and landlord are different, landlord is in an environment of competition, while peasants are in an environment of both competition and cooperation and thus have more to think about. Consequently, DQN-IRL makes no big difference in the role of up peasant, compared with DQN. Compared with NFSP, the winning rate of DQN-IRL is higher and the variances of them are close, which means that to some extent, DQN-IRL is more suitable for Douzihu than NFSP.

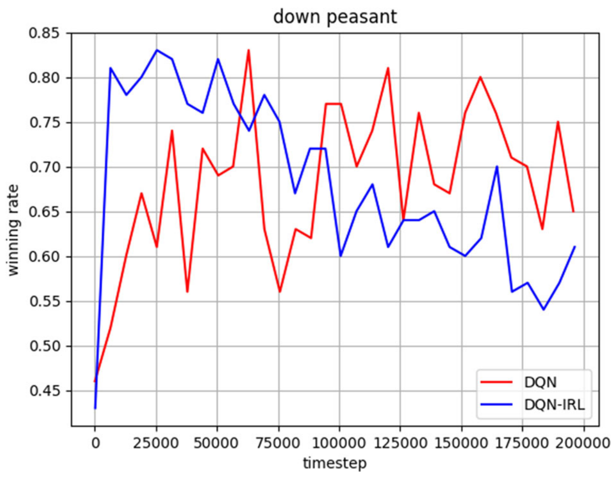

Fig. 8 DQN-IRL and DQN as down peasant

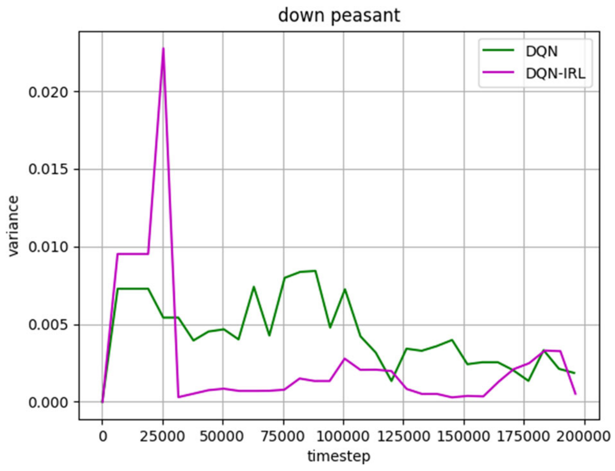

Fig. 9 The variance of DQN-IRL and DQN as down peasant

4 Discussion

By comparing the winning rate and variance of DQN-IRL and DQN algorithm, we can see that a small number of tracks bring obvious improvement to the algorithm, which shows that the effect of the algorithm can be improved by this method in dealing with the sparse reward problem existing in games like Doudizhu. According to the experimental results, compared with DQN algorithm, the performance of DQN-IRL in Doudizhu is improved in most cases, even though the winning rate of DQN-IRL is not as good as DQN in some cases resulting from the lack of training data. Nevertheless, compared with other methods to deal with the problem of hard sparse reward, the adoption of IRL in Doudizhu is more time and effort saving, and effectively alleviates the problem of sparse rewards. The application of IRL

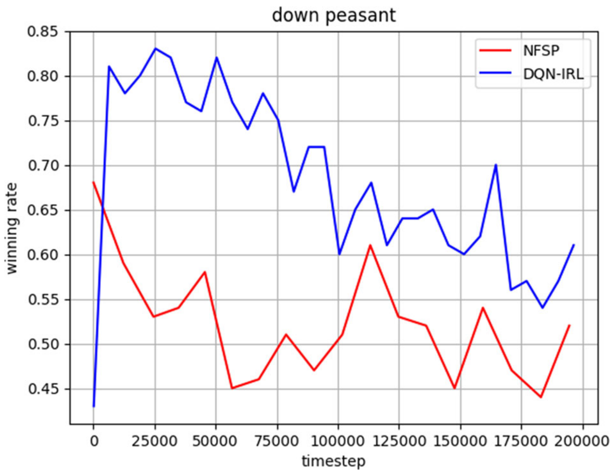

Fig. 10 DQN-IRL and NFSP as down peasant

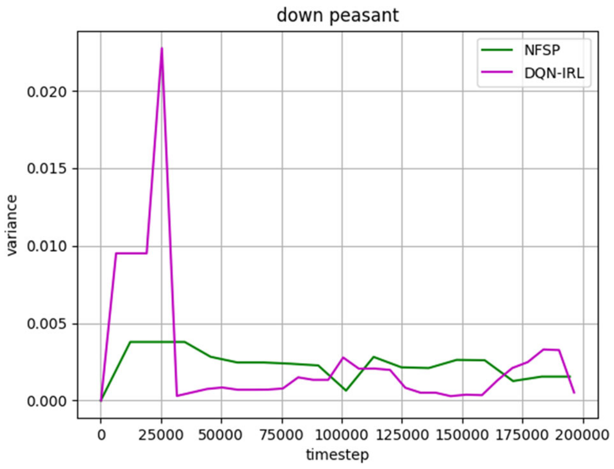

Fig. 11 The variance of DQN-IRL and NFSP as down peasant

in DQN algorithm can effectively improve the performance of such algorithms on sparse reward problems. Besides, due to the characteristics of IRL, even if training data (i.e., expert trajectories) are required for training, it does not need a large amount of data as support as other AI-based Doudizhu, which also makes our experiment require less time and easy to be reproduced.

Overall speaking, DQN-IRL is not highly complicated, the requirements for experimental equipment are not demanding, and the training speed is quite fast. In this framework, we only need to collect enough expert trajectories, feed the expert trajectories to the inverse reinforcement learning algorithm, and the reward function is obtained to be integrated into the DQN algorithm after a certain episode iteration. After enough iterations, the reward

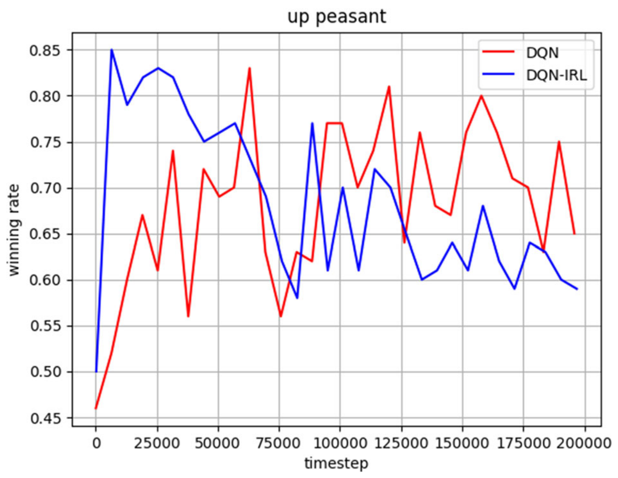

Fig. 12 DQN-IRL and DQN as up peasant

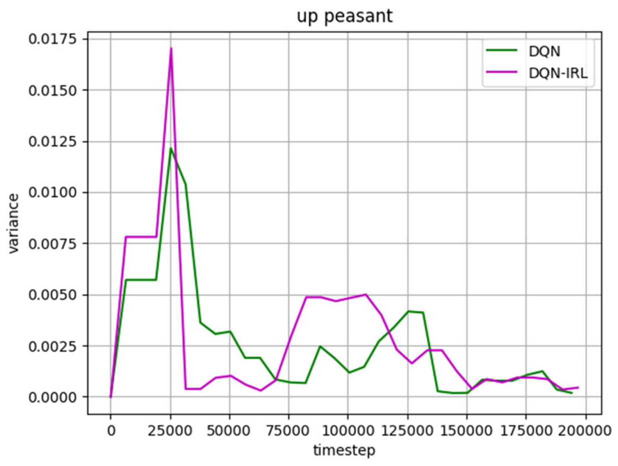

Fig. 13 The variance of DQN-IRL and DQN as up peasant

function in line with the expected goal can be trained under the framework of DQN, and the time complexity and space complexity are friendly.

5 Conclusion and Future Work

This paper focuses on the sparse rewards problem in Doudizhu and propose a framework DQN-IRL, which obtains desirable evaluation results in terms of both winning rate and performance stability, compared with one of the widely used baseline DQN. The desirable performance of DQN-IRL demonstrates the advantage of IRL when being used in sparse reward problems. When facing the fact that hard sparse reward problem is still difficult to be

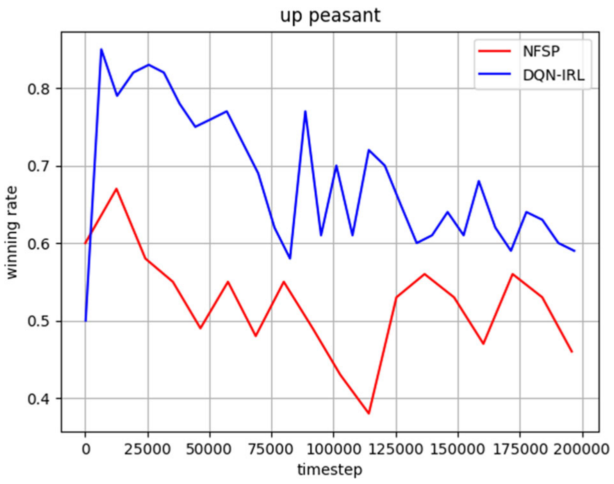

Fig. 14 DQN-IRL and NFSP as up peasant

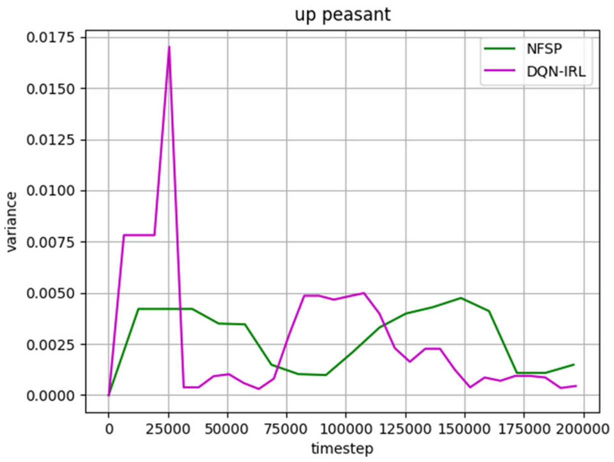

Fig. 15 The variance of DQN-IRL and NFSP as up peasant

solved after decoupling or stratification, we can choose IRL and deep reinforcement learning to build a reasonable learning framework to help the agent get better performance. This paper aims to verify the effectiveness of this framework in the face of the difficult sparse reward problem, and hopes to bring some inspirations to others, so as to build a more reasonable and excellent framework to deal with sparse reward problem.

Although our DQL-IRL framework performs well in dealing with the problem of sparse reward, there is still room for improvement. Firstly, on the basis of this framework, we can continue to add some deep learning methods, such as GRU(Gated Recurrent Unit) [26] network and dueling network [27]. GRU can further enhance the information capture capability of the agent in the game, dueling network can help the agent select the actions better, and weaken the adverse impact of the huge action space on the agent to a certain extent. However, these improvements based on neural networks will also greatly increase the demand for computing resources. We note that ML-ELM (multilayer extreme learning machines) are non-iterative and fast due to the random feature mapping mechanism [28]. Therefore, it may be better to try using ML-ELM method in this framework. Secondly, Doudizhu is an imperfect information game, it is important for player agents to learn to make predictions about the hand cards of teammates or opponents during the card playing, which is our next research direction. Thirdly, Doudizhu is a multi-agent game model, thus we also plan to study the different policies for agents from the aspects of both cooperation and competition in the future. Moreover, we have noticed that the winning rate of the same algorithm varies greatly in case of different positions which is in line with the rules of Doudizhu game in reality and is worth being studied. Last but not least, some researchers have proposed methods based on ELM(extreme learning machines) with better robustness and generalization [29]. And for reinforcement learning, robustness and generalization are quite important, so we will further explore the connection between EML and reinforcement learning.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by YK and HS. Code is modified by HS, XW and YR. The first draft of the manuscript was written by HS and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Code availability There are not available Code or data.

Declarations

Conflict of interests The authors have no relevant financial or non-financial interests to disclose.

Ethics Approval This is an observational study. The XYZ Research Ethics Committee has confirmed that no ethical approval is required.

Consent to Participate Informed consent was obtained from all individual participants included in the study.

Consent for Publication The authors affirm that human research participants provided informed consent for publication of the images in the paper.

References

1. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
2. Silver D, Schrittwieser J, Simonyan K et al (2017) Mastering the game of go without human knowledge. Nature 550(7676):354–359
3. Silver D, Hubert T, Schrittwieser J, et al. (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815
4. Machado MC, Bellemare MG, Talvitie E et al (2018) Revisiting the arcade learning environment: evaluation protocols and open problems for general agents. J Artif Intell Res 61:523–562
5. Vinyals O, Ewalds T, Bartunov S, et al. (2017) Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782
6. Browne CB, Powley E, Whitehouse D et al (2012) A survey of monte Carlo tree search methods. IEEE Trans Comput Intell Ai in Games 4(1):1–43
7. Brown N, Sandholm T (2019) Superhuman AI for multiplayer poker. Science 365(6456):885–890

8. Brown N, Sandholm T (2018) Superhuman AI for heads-up no-limit poker: libratus beats top professionals. Science 359(6374):418–424
9. Jiang Q, Li K, Du B, et al. (2019) DeltaDou: Expert-level Doudizhu AI through Self-play. IJCAI. pp 1265–1271.
10. You Y, Li L, Guo B, et al. (2019) Combinational Q-Learning for Dou Di Zhu[J]. arXiv preprint arXiv: 1901.08925
11. Zha D, Xie J, Ma W, et al. (2021) DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning. arXiv preprint arXiv:2106.06135
12. Zhang X, Wang H, Stojanovic V, et al. (2021) Asynchronous Fault Detection for Interval Type-2 Fuzzy Nonhomogeneous Higher-level Markov Jump Systems with Uncertain Transition Probabilities. IEEE Trans Fuzzy Syst, pp 1–1
13. Zxa B, Xla B, Vs C (2021) Exponential stability of nonlinear state-dependent delayed impulsive systems with applications. Nonlinear Anal Hybrid Syst, 42
14. Xin X, Tu Y, Stojanovic V et al (2022) Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems. Appl Math Comput 412(1–3):126537
15. Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations Theory and application to reward shaping. Morgan Kaufmann Publishers Inc., Burlington
16. Jaderberg M, Mnih V, Czarnecki WM, et al. (2016) Reinforcement Learning with unsupervised auxiliary tasks
17. Li S, Wang R, Tang M, et al. (2019) Hierarchical reinforcement learning with advantage-based auxiliary rewards
18. Kulkarni TD, Narasimhan KR, Saeedi A, et al. (2016) Hierarchical deep reinforcement learning. Integr Temp Abstract Intrinsic Motiv
19. Parr R, Russell S (1998) Reinforcement Learning with Hierarchies of Machines. In: Conference on advances in neural information processing systems. MIT Press
20. Abbeel P, Ng AY (2011) Inverse reinforcement learning. In: Webb GI, Sammut C (eds) Encyclopedia of machine learning. Springer, Boston MA
21. Mnih V, Kavukcuoglu K, Silver D, et al. (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602
22. Wu Z, Sun L, Zhan W et al (2020) Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. IEEE Robot Automation Lett 5(4):5355–5362
23. Abbeel P, Ng AY (2004) Apprenticeshipship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on Machine learning. 1
24. Zha D, Lai K H, Cao Y, et al. (2019) Rlcard: A toolkit for reinforcement learning in card games. arXiv preprint arXiv:1910.04376
25. Zhang L, Chen Y, Wang W et al (2021) A monte carlo neural fictitious self-play approach to approximate Nash equilibrium in imperfect-information dynamic games. Front Comput Sci 15(5):1–14
26. Cho K, Merrienboer BV, Gulcehre C, et al. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. Comput Sci
27. Wang Z, Freitas ND, Lanctot M (2015) Dueling network architectures for deep reinforcement learning. JMLR. https://doi.org/10.48550/arXiv.1511.06581
28. Zhang J, Li Y, Xiao W, et al. (2020) Non-iterative and fast deep learning: multilayer extreme learning machines. J Franklin Inst, 357(13)
29. Zhang J, Li Y, Xiao W, et al. (2020) Robust extreme learning machine for modeling with unknown noise. J Franklin Inst, 357(14)