Complex & Intelligent Systems
https://doi.org/10.1007/s40747-021-00345-6

ORIGINAL ARTICLE

A game strategy model in the digital curling system based on NFSP

Yuntao Han[1] · Qibin Zhou[1] · Fuqing Duan[1]

Received: 29 November 2020 / Accepted: 17 March 2021

Abstract
The digital curling game is a two-player zero-sum extensive game in a continuous action space. There are some challenging problems that are still not solved well, such as the uncertainty of strategy, the large game tree searching, and the use of large amounts of supervised data, etc. In this work, we combine NFSP and KR-UCT for digital curling games, where NFSP uses two adversary learning networks and can automatically produce supervised data, and KR-UCT can be used for large game tree searching in continuous action space. We propose two reward mechanisms to make reinforcement learning converge quickly. Experimental results validate the proposed method, and show the strategy model can reach the Nash equilibrium.

Keywords Game strategy · Digital curling · NFSP · Deep reinforcement learning

Introduction

For a long time, machine games and artificial intelligence have been closely related, and machine games are also an important form of artificial intelligence. From the game theory of von Neumann [1], the father of computers, to the well-known AlphaGo [2], today, machine games have always been in the public eyes. Game theory is an important guarantee for the study of machine games, and it has been widely used in the machine games.

Curling is a zero-sum Olympic sport played by two teams of players, with a corresponding strategy for each shot. The digital curling system simulates matches between two teams or AI programs to play curling through a 2D physics simulator, each team shoots eight balls in turn, and the team with the highest cumulative score wins the simulated match. Digital curling game has many action strategies, large search space and strong uncertainty, and it is a typical extensive form game [3]. The extensive form game is a tree-based form that expands in the form of multi-player interaction. For extensive form games, some challenging problems are still not solved well, such as the uncertainty of strategy, the large game tree searching, and the use of large amounts of supervised data, etc. To better solve decision-making problem in extensive form games, Heinrich and Silver [4] proposed the neural fictitious self-play (NFSP) method which combines deep neural network and fictitious self-play (FSP) [5]. This method can effectively solve the problem of large amounts of supervised data. NFSP Players use deep Q learning to approximate their own best response estimates, supervise historical optimal strategies, and constantly update the average strategy. Without any prior knowledge, the system can approximately converge to the Nash equilibrium [6]. However, when NFSP is in a relatively large search space and for a large extensive form game, the performance is not good.

Monte Carlo tree search (MCTS) methods, such as Upper Confidence Bounds Applied to Trees (UCT) [7] and the random discrete action set [8] are used to solve the continuous action space problem, and accurate evaluation is carried out through deeper exploration and better selection. However, random uncertain choices may omit good strategies. To solve the problem of uncertainty and incompleteness in the choice of continuous action space strategies, Timothy Yee [9] proposed KR-UCT (Kernel Regression UCT), the algorithm can be applied to the whole continuous action space, considering the action quality information and the unexplored space part. For digital curling games, some researchers used Monte Carlo search and KR-UCT algorithms to improve the game performance. However, these solutions still need a lot of supervised data and prior knowledge. The prior knowledge usually limits the scalability of the game, and it is also difficult to acquire a large amount of supervised data for machine games.

✉ Fuqing Duan
fqduan@bnu.edu.cn

1 College of Artificial Intelligence, Beijing Normal University, Beijing 100875, People's Republic of China

Published online: 31 March 2021

In this work, we combine NFSP and KR-UCT for digital curling games, where NFSP can avoid manual labeling of supervised data, and KR-UCT can be used for large game tree searching in continuous action space. We also propose two reward mechanisms for the deep reinforcement learning, i.e. situation evaluation reward in each round and a future reward, and we analyze pros and cons of the two reward mechanisms in the experiment. Moreover, we use the exploitability metric to validate whether the method can converge to Nash equilibrium, which has not been used in previous digital curling systems.

The remainder of the paper is organized as follows. "Related works" introduces related work. "Methods" provides the details of the proposed method. "Experiments" presents experimental results. Finally, we give some conclusions.

Related works

Digital curling games

Early research mainly focuses on the motion model of the curling sport, which can provide a guidance for the game play. Maeno et al. [10] proposed a numerical model based on the evaluation index of friction amplification coefficient, and they established a dynamic model by studying the friction relationship between the bottom of the curling stone and the ice surface. Lozowski et al. [11] further validated the influence of thermodynamic friction, velocity of motion, angular velocity, etc. on the trajectory and numerical model of curling stone motion through an IMU-based instrument curling handle experiment.

Some researchers [12,13] have developed digital curling systems (ITO) for curling competition. To obtain a better game state assessment and try to overcome the influence of uncertainty conditions on the results, Yamamoto et al. [12] proposed an evaluating function based on probability distributions. This method can quickly find an optimal strategy under the same conditions. Yamamoto et al. [14] proposed a static evaluation function based on a deep neural network model and developed a new digital curling system called Jiritsu-nn, which improved the inflexibility of the system. To facilitate computer programs to process curling competition strategies, it is necessary to record curling stone movement images. Hong et al. [15] conducted an in-depth study on changing the camera attitude to obtain curling images. Won et al. [16] developed a curling robot based on curling strategies, which defeated the top-ranked amateur team in Korea.

In the digital curling system, all actions and state spaces are continuous, and each curling shot is very random. Katsuki Ohto et al. [17] and Ahmad et al. [18] proposed a Monte Carlo tree based action decision method and a Delaunay triangulation-based method respectively for the continuous action state space, both of which can select better strategy and achieve higher scores in curling competitions. To deal with the uncertainty of the sample actions, Yee et al. [9] proposed an improved algorithm (KR-UCT) combining Monte Carlo trees and kernel regression, which improved the correct selectivity of action strategies by 12% compared to other algorithms in the international digital curling competition. Also for the uncertainty of random action decision, Ahmad and Zaheen [19]studied last shot. They used Delaunay sampling to enable an 80% reduction in computational time for the correct strategy under the same conditions, greatly improving computational efficiency.

To reduce the computation cost for large-scale game tree searching, Lee et al. [20] proposed a deep reinforcement learning framework in the continuous action state space, which contains a value network for evaluating curling positions and a strategy network for selecting actions, improving the win rate by 13% in the international digital curling competition. However, their method requires a lot of supervised data and prior knowledge, which limits the scalability of the game.

NFSP

NFSP [4] uses two neural networks to fit the optimal strategy and the average strategy. One network is a deep reinforcement learning network that approximates the optimal response; the other network is a supervised network that uses supervised learning to train its past average strategy. At the beginning, the circular buffer and the reservoir were established to store the optimal strategies and the average strategies. These two data sets are used to train the deep reinforcement learning network and the supervised network respectively. During the game, the agent chooses actions from the mixed strategy, and chooses the best response action and the average strategy action with probability. When the agent performs each action and strategy, it will get the next reward value and state information. The agent records the current state, action and the next reward state. If the action selected by the agent is the best response, the observed state and action are stored, and optimize the model by updating network parameters.

In NFSP, two techniques are adopted, i.e. reservoir sampling and anticipatory dynamics. Reservoir sampling is used to store the state, action, and reward of the best response in each game. Each action is selected in a mixed strategy, which may not be the best strategy, and only the best response strategy is stored. Reservoir sampling can avoid window artifacts caused by memory sampling. and helps agents achieve synchronous self-learning. Anticipatory dynamics enables the agent to predict the approximate value of the opponent's average strategy in a short time. The agent will generate supervised data for the supervised network. In a fictitious game, the agent makes the best response to the opponent's strategy in a short continuous dynamic time, which makes the model more stable. NFSP is not fit for a large extensive form game with a continuous action space.

Methods

Curling is an extensive-game. The core of the extensive form game is a game tree with a root node. This tree is composed of root nodes and leaf nodes. From one node to another is the action strategy selected by the player according to the state. Each non-terminal node (non-leaf nodes) has a corresponding player to make a corresponding action or strategy. Each terminal node (leaf node) has the final profit for the player. Multiple players take turns to make action strategies until the end state is reached, and the game tree is constructed successfully.

The extensive form game is composed of elements. A series of states of the player are represented by s, which also includes the initial state of the game ϕ. The terminal state of the game is represented by z, which is the leaf node in the game tree. $R_{s_t}^{a_t}$ express the action a_t taken in state s_t. R and Q represent the utility values, which represent the reward values obtained after taking strategies.

Overview

To avoid manual labeling of the large quantity of supervised data in digital curling games, we use NFSP for adversary training of the strategy model. NFSP includes two networks. The first network is a deep reinforcement learning strategy network β_i that fits the extensive game, and the other network is a supervised learning network π_i. The following describes in detail how to use deep reinforcement learning to perform fictitious self-game training in the extensive game and how to generate supervised data.

Deep reinforcement learning and supervised learning use the same convolutional neural network architecture as shown in Fig. 1. The information in the curling house can be discretized into 32*32 position points as the state input S to the deep reinforcement learning and supervised learning network. The input information also includes own curling position information, opponent curling position information, and the number of throwing curling, and a total of 29 dimensions are abstracted. The network output is the value of the probability distribution obtained by the KR-UCT discretization sampling with 32*10*3 (32*10 refers to the angle and strength of the hitting ball, while 3 refers to the left-turn, right-turn or non-spin ball during hitting). In the network design, the pooling layer is not used to extract key information, but all the information is retained for processing, using a 7-layer convolutional layer network. The final output is the probability distribution of 960-dimensional actions.

Strategy network uses deep reinforcement learning

The curling game is an extensive game. The sente and gote game strategies are completely different. Two deep reinforcement learning networks are used to learn the sente and gote game strategies separately.

The value of the strategy value function $Q_\pi(s_t, a_t)$ is fitted by the neural network, the parameters θ in the network need to be learned iteratively. The reward is obtained through the agent interacting with the environment. Since the curling game is an extensive game, the final reward cannot be obtained immediately after the action is made. Each time the decision maker can only evaluate the value of the current state when making a decision, that is, the game state becomes $s_t{}'$ after the decision maker selects action a_t in the state s_t. In the $s_t{}'$ state, the decision maker cannot choose an action. At this time, the opponent needs to choose an action. Therefore, the general reinforcement learning strategy value, which is as Eq. (1), cannot be used to update the network parameters.

$$Q_\pi(s_t, a_t) = R + \gamma * \text{argmax}x_a Q_\pi(s_t{}', a_t), \quad (1)$$

where γ is reinforcement learning attenuation factor.

The opponent's average strategy is crucial for the decision maker's strategy evaluation. After the decision maker performs action a_t to reach the state $s_t{}'$, the opponent chooses the average strategy a^{opp} to move to the state s_{t+1}. The decision maker selects the next action in the state s_{t+1}. The network update method of deep reinforcement learning is that after the decision maker takes action a_t in state s_t, the opponent takes action a_t^{opp}, and the reward obtained at this time is $R_{s_t}^{a_t a_t{}^{\text{opp}}}$. The return for the current environment is $V(s)$, then the strategy value function is

$$Q_\pi(s_t, a_t) = R_{s_t}^{a_t a_t{}^{\text{opp}}} + \gamma V(s_{t+1}) \quad (2)$$

Supervised network

The supervised network is to train the average strategy, where supervised data is generated during the reinforcement learning sampling process. If the reward obtained after each selected action is greater than a certain threshold ε, the action is considered to be the best response $BR_S(a)$ in the current state s. This state and action pair (s, a) is stored as a piece of supervised learning data and used to train the average strategy of the supervised network. As the number of training increases, the exploitability of the strategy will become lower and lower, and the average strategy will get closer and closer to the Nash equilibrium strategy.

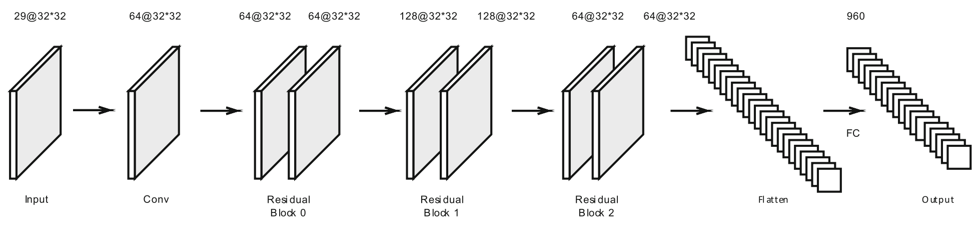

Fig. 1 Network architecture

At the beginning of the supervised network model training, the opponent's strategy is relatively poor, and the attenuated learning rate cannot be used directly for learning. As the number of training increases, the opponent strategy gets better and better, and then the dynamic attenuated learning rate is set to reduce the loss of the network. The supervised learning network can be regarded as a function SL(S), the sampling stochastic gradient descent method is used to train the network, and its loss function is

$$\mathrm{Loss}_{SL} = \|SL(S;\theta) - BR_S(a)\| + c|\theta|^2, \tag{3}$$

where c is the regularization parameter.

Reward

There is no immediate reward after choosing an action in an extensive game. The factual regret value can effectively solve this problem. The factual value of the middle node of the game tree is estimated when the game leaf node is not reached. We use two reward functions for strategy evaluation.

The first reward is to perform cumulative evaluation (SER situation evaluation reward) in each round according to the stone distribution of the decision maker and the opponent in the house. The flaw of this evaluation is that each turn is evaluated as an independent game, but in fact they are related. For example, the 8th round of curling is more effective than the 1st round. We introduce the weight of the attenuation coefficient $w(t) = \frac{1}{\ln(c-t)}$, where c is a constant and t is the current round number. The deep reinforcement learning network is RL(S), then its loss function is

$$\begin{aligned}&\mathrm{Loss}_{\mathrm{RL}} = \left\| Q_{\mathrm{estimate}}(s_t, a_t) - Q_{t\arg et}(s_t, a_t) \right\| \\ &Q_{\mathrm{estimate}}(s_t, a_t) = \mathrm{argmax}_{a_t} RL(S) \\ &Q_{\mathrm{target}}(s_t, a_t) = R_{s_t}^{a_t a_t{}^{\mathrm{opp}}} + \gamma * \mathrm{argma}x_{a_{t+1}} Q_{\mathrm{constant}}(s_{t+1}, a_{t+1})\end{aligned} \tag{4}$$

The second reward function is future reward (FR). The decision maker chooses the last action. After the game is over, the $R_{s_t}^{a_t a_t{}^{\mathrm{opp}}}$ value is the score of the decision maker's win or lose. If the game is not over after the action is selected, the value of $R_{s_t}^{a_t a_t{}^{\mathrm{opp}}}$ is 0. This reward method is biased toward learning to choose higher reward actions when the game is not over.

Adversarial training by fictitious self-game

For training strategy model, the deep reinforcement network and supervised network are trained in adversarial way to learn the best response strategy and average strategy of the sente and gote respectively. In each round of shots, the decision maker selects action a_t according to the best strategy in the current state s_t. and then simulates the opponent's average strategy selecting action a_t^{opp} to reach the state s_{t+1}, and obtain the reward for environmental feedback. Afterwards, the decision maker continues to choose the best strategy action in the s_{t+1} state until the end of the game. The specific algorithm flow is described as Algorithm 1. The agent uses the better strategy in deep reinforcement learning as the supervised data to train the supervised network, and gradually finds the Nash equilibrium strategy in the process of continuous gaming.

Experiments

Experimental parameter settings

During the training process, the (s, a) label whose reward obtained by taking action a in state s is greater than the threshold ε is stored. After 512 pieces of data are accumulated, they are used as training data for supervised network learning. The batch size is 512. According to the decision makers strategies and the opponents average strategy, we sample 100 actions to estimate the reward for taking action a in the current state. The learning rates of reinforcement learning and supervised learning are set to 0.00025 and 0.001 respectively. All the optimization methods of the model adopt stochastic gradient descent. Reinforcement learning attenuation factor $\gamma = 0.99$, action reward range [− 1, 1], optimal response action screening threshold $\varepsilon = 0.4$. The regularization parameter c is $2e^{-5}$.

Algorithm 1 Our Method in each iteration

```
1: $RL^{sente}$ ⇐ the policy-value network of sente player
2: $SL^{sente}$ ⇐ the average strategy network of sente player
3: $RL^{gote}$ ⇐ the policy-value network of gote player
4: $SL^{gote}$ ⇐ the average strategy network of gote player
5: $s_t^{sente}$ ⇐ the current state(need sente player to take action)
6: $sit_t^{sente} = SER(s_t^{sente})$
7: $brs^{sente}$ ⇐ best response actions for sente in specific state
8: $brs^{gote}$ ⇐ best response actions for gote in specific state
9: while $s_t^{sente}$ is not end do
10:     $a_t^{sente} = RL^{sente}(s_t^{sente})$
11:     $s_t^{gote} = takeAction(s_t^{sente}, a_t^{sente})$
12:     $sit_t^{gote} = SER(s_t^{gote})$
13:     $a_t^{gote} = SL^{gote}(s_t^{gote})$
14:     $s_{t+1}^{sente} = takeAction(s_t^{gote}, a_t^{gote})$
15:     $sit_{t+1}^{sente} = SER(s_{t+1}^{sente})$
16:     $a_{t+1}^{sente} = RL^{sente}(s_{t+1}^{sente})$
17:     $s_{t+1}^{gote} = takeAction(s_{t+1}^{sente}, a_{t+1}^{sente})$
18:     $sit_{t+1}^{gote} = SER(s_{t+1}^{gote})$
19:     $Reward_t^{sente} = sit_{t+1}^{sente} - sit_t^{sente}$
20:     $Reward_t^{gote} = sit_{t+1}^{gote} - sit_t^{gote}$
21:     UPDATERL($RL^{sente}$,$s_t^{sente}$,$a_t^{sente}$,$s_{t+1}^{sente}$,$a_{t+1}^{sente}$)
22:     UPDATERL($RL^{gote}$,$s_t^{gote}$,$a_t^{gote}$,$s_{t+1}^{gote}$,$a_{t+1}^{gote}$)
23:     if $Reward_t^{sente} > \epsilon$ then
24:         $brs^{sente}.Add(pair(s_t^{sente}, a_t^{sente}))$
25:     end if
26:     if $Reward_t^{gote} > \epsilon$ then
27:         $brs^{gote}.Add(pair(s_t^{gote}, a_t^{gote}))$
28:     end if
29: end while
30: if $brs^{sente}$ is not empty then
31:     UPDATESL($SL^{sente}$,$brs^{sente}$)
32: end if
33: if $brs^{gote}$ is not empty then
34:     UPDATESL($SL^{gote}$,$brs^{gote}$)
35: end if
```

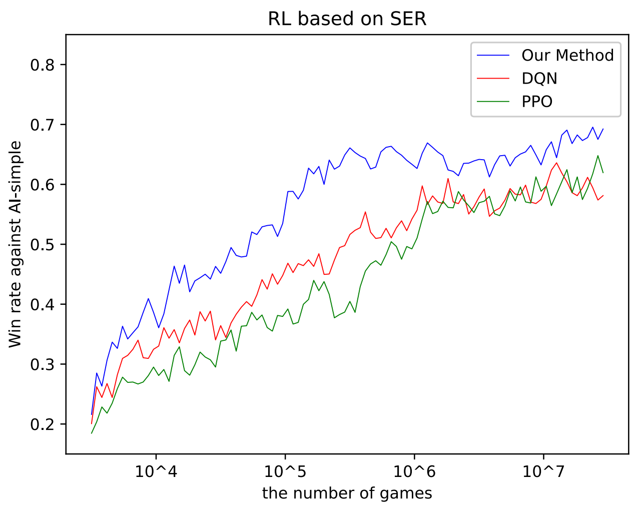

Fig. 2 Win rate against AI-simple based on SER

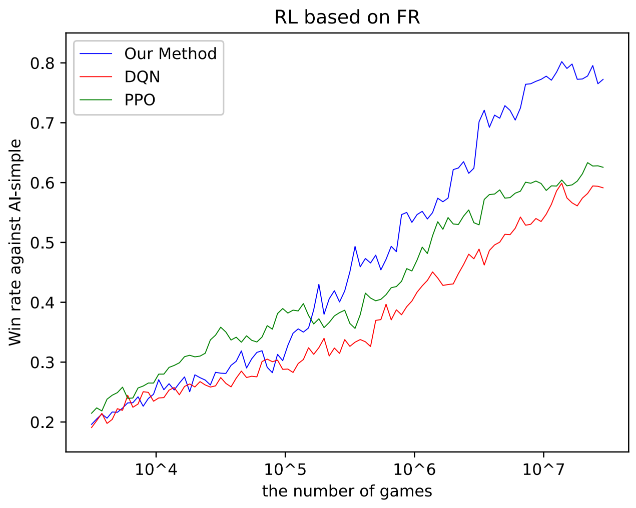

Fig. 3 Win rate against AI-simple based on FR

Results and discussion

We used two different reward functions SER and future reward to train two classic reinforcement learning models for experimental comparison, i.e. DQN (Deep Q-learning) model based on strategy value [21] and PPO (Proximal Policy Optimization) model [22]. DQN and PPO models are widely used in Atari games and machine games, and have achieved good results. DQN learns control strategies from raw data through end-to-end convolutional neural networks. PPO optimizes an objective function by using multiple epochs stochastic gradient ascent. These models and our method are tested against AI-simple in the international digital curling competition. Each adversarial test consists of two rounds of games. The sente and gote players take turns in one round. The scores of the two rounds are added together, and the highest scorer wins.

Figures 2 and 3 show the variation of the winning rate of the three models by the game number. We can see that our method has a higher winning rate against AI-simple than the DQN model and PPO model at the same number of training iterations. The convergence speed of our method is faster than the other two models. It is because our method adopts adversary learning to learn higher reward actions. The supervised data in the supervised network become better and better along with the adversary learning. DQN and PPO agents need to explore all actions, and evaluate the reward of each action and the probability distribution of all actions.

From Figs. 2 and 3, we can see that the method based on the SER reward function converges faster than future reward during the training process. At about 500,000 iterations of training, our method's (SER) win rate against AI-simple is basically stable at 64.4%, while the future reward-based method requires about 1.5 million iterations of training to achieve 65.1%. The reason is that in each round of curling, there is a lot of action choices, and most of the choices have very poor reward. SER makes the selected action immedi-

Table 1 Winning percentages that our method beats existing programs

Program	Winning percentage
KR-DL-UCT	61.4%
KR-UCT	83.2%

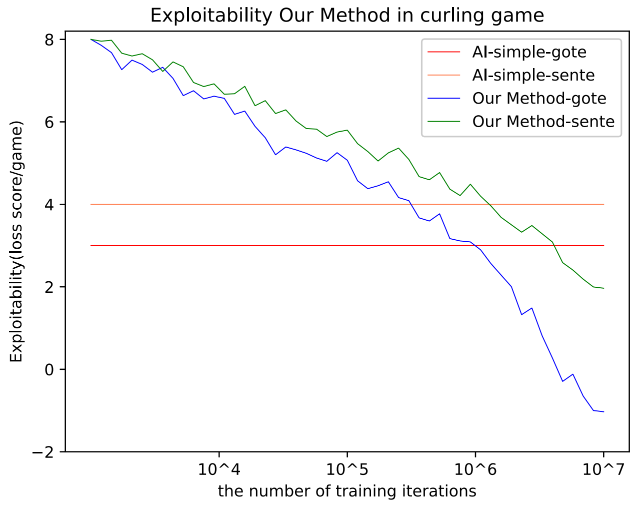

Fig. 4 Exploitability

ately obtain a higher reward, similar to the greedy learning strategy. The future reward of actions selected in each round will be biased toward the maximum future benefits and learn to find the best overall strategy. However, although the SER-based (SER-RL) converges faster, the final strategy win rate is 69.2%. On the contrary, the Future reward-based reinforcement learning (FR-RL) has a slow convergence rate, but it can eventually learn the overall situation. The strategy have a winning rate of 79.1%.

We also compared the performance of our method with existing programs, KR-UCT [9] and KR-DL-UCT [20]. Table 1 shows the winning percentages that our method beats existing programs. There are 1000 games in each match (500 sente games and 500 gote games, considering the gote game has a certain advantage). From the table, we can see that the program trained by our method has better performance than KR-DL-UCT and the vanilla KR-UCT.

To validate the strategy learned by our method can converge to Nash equilibrium. we determine the exploitability of strategies using the factual regret value, that is, the lower the reward of the opponent is when the decision maker choose the action, the lower the exploitability of the strategy learned by the decision maker, and the closer the strategy is to the Nash equilibrium strategy. Figure 4 shows the variation of the exploitability by the training iterations. It can be seen that the exploitability of AI-simple's sente and gote hits are 4 and 3 points respectively. With the increase of the number of training iterations, the exploitability of our method becomes lower and lower. When the training iteration is 10,000,000 rounds, the exploitability of the gote strategy of this method is − 1.03 points, and the exploitability of the sente strategy is 1.97 points. The reason is that the gote has superiority over the sente, and it is reasonable that the exploitability of the gote strategy is negative. The experimental results show that our method can learn the Nash equilibrium strategy of curling game as the number of training increases.

Conclusion

In this work, we consider the digital curling game as a two-player zero-sum extensive game in a continuous action space. We combine KR-UCT and NFSP for digital curling games, which can avoid manual labeling of supervised data. The agent uses two adversary learning networks to effectively solve the Nash equilibrium strategy of the curling games, where a deep reinforcement learning network generates the supervised data to train a supervised network to learn average strategy. To make reinforcement learning converge quickly, we propose two reward functions for the reinforcement learning. The trained model competes with the AI-simple program to achieve a higher winning rate in the international digital curling competition. We use the factual regret value to determine the exploitability of the strategy, and use the exploitability metric to validate that the model can reach the Nash equilibrium.

Acknowledgements This work is supported by National Key Research and Development Project Grant, grant/award number: 2018AAA01008-02.

Author Contributions YH presents the idea. QZ does experimental design. Prof FD contributes the overall article structure and method optimization.

Funding This work is supported by National Key Research and Development Project Grant, grant/award number: 2018AAA0100802.

Availability of data and material No data and material are needed in this work.

Declarations

Conflict of interest No conflict.

Code availability Code will be available after paper publication.

References

1. Roughgarden T (2010) Algorithmic game theory. Commun ACM 53(7):78–86
2. Wang FY, Zhang JJ, Zheng X et al (2016) Where does AlphaGo go: from church-turing thesis to AlphaGo thesis and beyond. IEEE CAA J Autom Sin 3(2):113–120
3. Zinkevich M, Johanson M, Bowling M et al (2007) Regret minimization in games with incomplete information. Adv Neural Inf Process Syst 20:1729–1736
4. Heinrich J, Silver D (2016) Deep reinforcement learning from self-play in imperfect-information games. arXiv preprint arXiv:1603.01121
5. Heinrich J, Lanctot M, Silver D (2015) Fictitious self-play in extensive-form games. In: International conference on machine learning. pp 805–813
6. Maskin E (1999) Nash equilibrium and welfare optimality. Rev Econ Stud 66(1):23–38
7. Kocsis L, Szepesvri C (2006) Bandit based monte-carlo planning European conference on machine learning. Springer, Berlin, pp 282–293
8. Dulac-Arnold G, Evans R, van Hasselt H et al (2015) Deep reinforcement learning in large discrete action spaces. arXiv preprint arXiv:1512.07679
9. Yee T, Lisy V, Bowling MH, Kambhampati S (2016) Monte Carlo tree search in continuous action spaces with execution uncertainty. In: IJCAI. pp 690–697
10. Maeno N (2014) Dynamics and curl ratio of a curling stone. Sports Eng 17(1):33–41
11. Lozowski E et al (2016) Comparison of IMU measurements of curling stone dynamics with a numerical model. Procedia Eng 147:596–601
12. Yamamoto M, Kato S, Iizuka H (2015) Digital curling strategy based on game tree search. In: 2015 IEEE conference on computational intelligence and games (CIG). IEEE, pp 474–480
13. Ito T, Kitasei Y (2015) Proposal and implementation of digital curling. In: Proceedings of the IEEE conference on computational intelligence and games, CIG, 469C473
14. Yamamoto M, Kato S, Iizuka H (2018) Learning of expected scores distribution for positions of digital curling. In: Proceedings of workshop on curling informatics (WCI2018). pp 8–9
15. Myungpyo H, Yoon KK, Sanghoon S (2018) Camera pose estimation based on concentric circles and parallel lines of a curling sheet, WCI2018, 12–15
16. Won D-O et al. (2018) Curly: an AI-based curling robot successfully competing in the olympic discipline of curling. In: IJCAI
17. Ohto K, Tanaka T (2017) A curling agent based on the Monte-Carlo tree search considering the similarity of the best action among similar states advances in computer games. Springer, Cham, pp 151–164
18. Ahmad ZF, Holte RC, Bowling M (2016) Action selection for hammer shots in curling. In: IJCAI. pp 561–567
19. Ahmad ZF (2017) Action selection for hammer shots in curling: optimization of non-convex continuous actions with stochastic action outcomes
20. Lee K, Kim SA, Choi J et al (2018) Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In: International conference on machine learning, pp 2937–2946
21. Mnih V, Kavukcuoglu K, Silver D et al. (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602
22. Schulman J, Wolski F, Dhariwal P et al. (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347