A System of DouDiZhu Based on Deep Q Learning

1st Lian Yuehan
181002222

2st Zhang Haina
181002616

3st Li Zhicheng
181002413

4st Fei Die
181002120

Abstract

The history of card games can be traced back to more than 2000 years ago in ancient China. DouDiZhu, invented in China around the 1990s, is a classic three-player cooperative game with incomplete information. Nowadays, DouDiZhu is very popular with all ages. We developed a system of DouDiZhu based on Deep Q Learning, improving the experience of players. And we analyzed the winning rate of each player in the three models to compare the algorithmic efficiency.

Index Terms

DouDiZhu, card, game, Deep Q Learning, incomplete information

I. Introduction

The history of card games can be traced back to more than 2000 years ago in ancient China. They spread all over the world and have been popular with all ages. DouDiZhu, Texas poker and Baccarat are the three most popular card games in the world.

DouDiZhu, invented in China's Hubei province around the 1990s, is a classic three-player cooperative game with incomplete information. Its name comes from the land reform movement in the early days of the founding of the People's Republic of China. DouDiZhu is attractive to the youth and the old for exercising their brains.

The classic DouDiZhu game involves three players and 54 cards in all. One player acts as a landlord and has the advantage of receiving three extra cards, the other two players play as peasants. The peasants need to cooperate against the landlord and will usually not beat each other. The peasants win if any of them is the first to play out all his cards in valid combinations, otherwise the landlord wins. The profits or losses in the game are shared between the peasants while the landlord carries by himself.

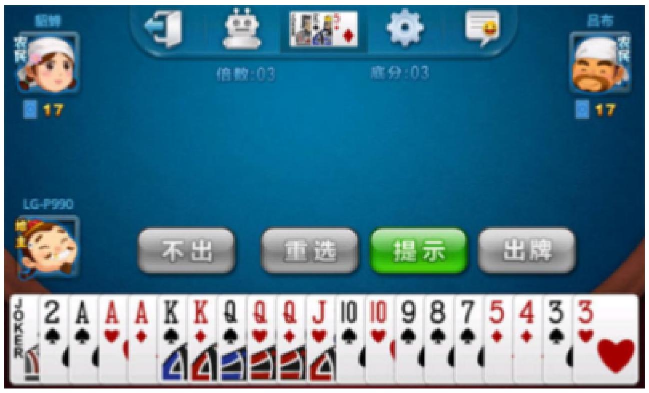

Fig. 1. DouDiZhu

DQN is actually a combination of deep learning and reinforcement learning knowledge, using deep networks framework to approximate the Q value in reinforcement learning.

DQN is composed of two networks. One is evaluation network, which is used to predict the Q value of different actions. The other is target network, which is used to simulate the real Q value.

II. Design

A. Interface design

We used Cocos2d-x4.0 as the foundation for our interface development. Cocos2d-x4.0 as an open source mobile 2D gaming framework released under the MIT license, is a cross-platform framework with Cocos2d-x4.0. The framework is compatible with C or Lua and can easily be deployed on mobile operating systems such as iOS, Android, BlackBerry or desktop operating systems such as Windows, Mac and Linux. This also leaves more possibilities for our subsequent development.

We built a simple and beautiful interface with online public and free landlord material and other images. The interface is composed of menu items, main interface items, game interface items, and end interface. At the time of the initialization of the program, we generated the basic cards, avatar and other related objects.

B. Game Flow and Rules

1) Game Flow: The game flow of our DouDiZhu platform based DQN is as follows. Firstly, click "Start" to enter the game. After dealing cards for three players, players choose whether to act as the landlord. The landlord needs to play a single card or any legal combination first. Then each subsequent player in anticlockwise order must either pass or beat the previous play by playing a higher combination of the same number and type of cards .

Once a player has no cards, game is over. The game scene will display the result of this time and input it to the database of DQN. Finally, click ”Close” to exit or choose ”Once Again” to continue.

According to the test needs, we modify the player types in the system, such as people, fixed rules, similar algorithm, DQN algorithm, etc.

2) The Data Inputting Flow of DQN: We have designed a way to transfer cards into the data for building the machine learning model. Firstly, decompose the current player's cards and find out all possible card combinations based on current player cards and previously played cards. Then input these possible card combinations to train the model, and the model will output the cards the current player needs to play.

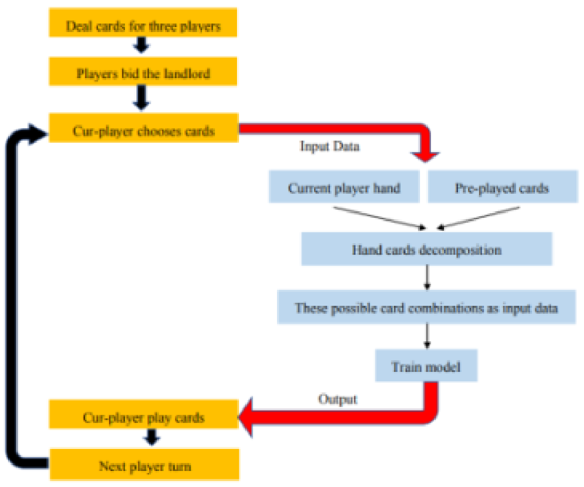

Fig. 2.

3) Playing Cards Rule: DouDiZhu consists of 52 standard cards, red joker and black joker. The cards rank from high to low: R(Red Joker), B(Black joker), 2, A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3. Each rank of standard card has 4 cards.

The landlord is first to play a single card or any legal combination. Each subsequent player in anticlockwise order must either pass (play no card) or beat the played cards by playing a higher combination of the same number and type of cards.

The landlord wins if he has played out all his cards in valid combinations first, otherwise the peasants win.

Valid card combinations are shown in Table 1.

Players should follow the rules to play the different card combination:

- Rocket is the highest rank card.
- Bomb is ranked lower than Rocket, but higher than any other card combination. If you want to compare the rankings of two bombs here, you should rank based on the combination of single cards.
- Except for Rocket and Bomb, comparison can only be performed within the same card combination based on the single card combination rank.

4) The Calling Relationship of Each Function in the Game Scene: Among them, the green line represents the scene conversion, and the pink line represents the call of custom functions.

The contents of the solid boxes are the function name, and the contents of the dotted boxes are the function explanation.

C. Card Decomposition

Before feeding the data for the training model, we need to find all possible card combinations that could be played. So, card decomposition is an essential part of this project.

Card Decomposition Pseudocode

```
status Judge_Pattern(type cards){
status card_pattern
unsigned int length = cards.number
for i:0->length-1
card_pattern.push_back(cards[i])
if 0 < cards.number&&
cards.number< 5
if cards[0]==cards[length-1] return Bomb;
if length==2&&cards[0]==Black_Jack
    &&cards[1]==Red_Jack
    return Rocket
if length==4 &&
    (cards[0]==cards[2]||cards[1]==cards[3])
    return Triplet_with_an_attached_card
if cards.number>=5
    if cards is continuous
    && cards[i] less than 2
     return Sequence
if cards.number>=6
    && cards.number % 2==0
for i:0->length-1 i+=2
    if(cards[i]!=cards[i+1])
       is_all_pair =false
       break
for i:0->length-1 i+=2
         vector.push_back(cards[i])
     if vector is continuous
     return Sequence of pairs
for i:0->length-1
   if i+1<length && cards[i]==cards[i+1]
     if i+2<length && cards[i]==cards[i+2]
       if i+3<length && cards[i]==cards[i+3]
       Bomb++;
      else Triplet++,i+=3
        else Pair++,i+=2
    else Single++,i++}
```

We prepared the function to check every card suit: single suit, pair suit, triple suit, bomb suit, rocket suit, and so on. Then use the get-total-moves function to find all possible card combinations and save them in the corresponding card type dictionary separately. Then based on the previously played cards type, find out all card combinations to beat the previously played cards.

TABLE I
CARD COMBINATION DESCRIPTION

Type	Brand	Example
Single	one card	6
Pair	two cards of the same rank	6-6
Triplet	three cards of the same rank	9-9-9
Triplet with an attached card	a triplet with a different single card	6-6-6-8
Triplet with an attached pair	a triplet with a pair	Q-Q-Q-6-6.
Sequence	at least five cards of consecutive rank. Jokers cannot be used	8-9-10-J-Q
Sequence of pairs	at least two triplets of consecutive ranks from 3 to A. 2 cannot be used	4-4-4-5-5-5
Sequence of triplets with attached cards	an extra card is added to each triplet. Only the triplets have to be in sequence	7-7-7-8-8-8-3-6.
Sequence of triplets with attached pairs	an extra pair is attached to each triplet	8-8-8-9-9-9-4-4-J-J
Bomb	four cards of the same rank	8-8-8-8
Rocket	a pair of jokers	

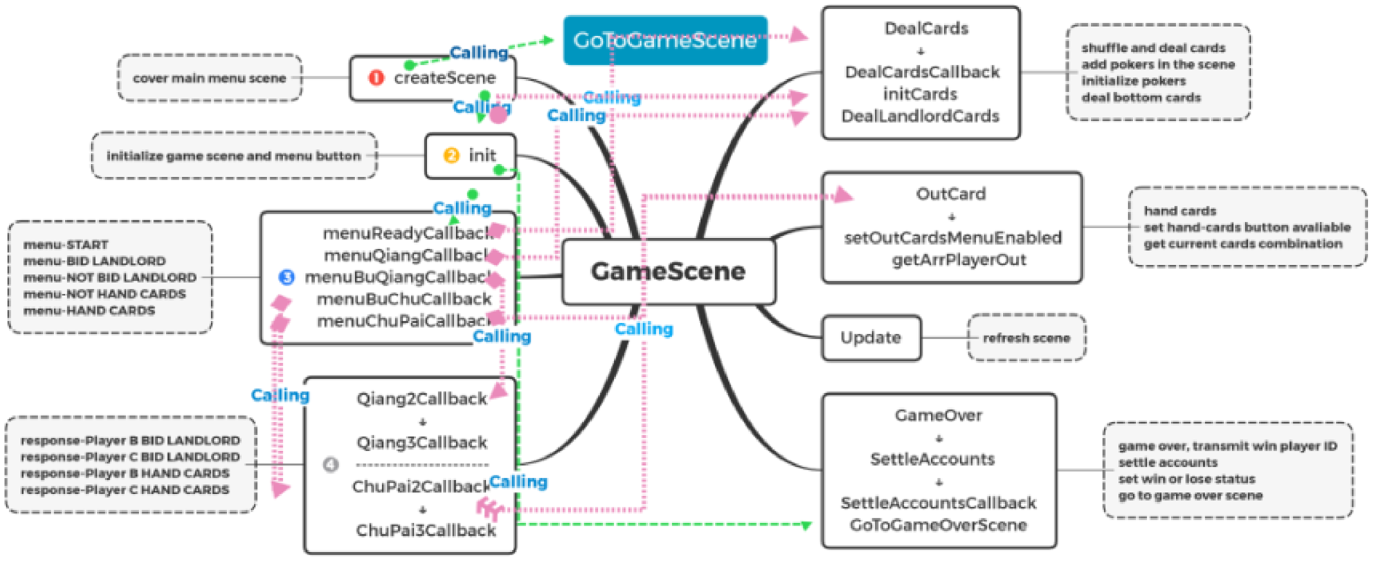

Fig. 3.

D. The Calculation of Patterns Value and the Following Principle

The calculation of pattern value is to digitize its valuation and provide data for agents to adopt strategies. The traditional static method of pattern value calculation uses an accurate score to evaluate each combination of cards. Its advantage is easy for the agent to understand and identify. In the design of pattern value, we set different scores for different card types. Because comparison in DouDiZhu depends on card types, the traditional scores will lead to inaccurate results.

This paper introduces the idea of probability to solve the problem of inaccurate card value. Two probability indexes are added into the calculation of each combination card: the first is the probability $p(i)_{strength}^{state}$ that the combination card i will be followed in the current game; the second is the probability $p(i)_{strength}^{statistics}$ strength that the combination card I will be managed after the occurrence of each combination card i.

The calculation formula of the pattern value expressed by strength is as follows:

$$V(i)_{strength} = \alpha p(i)_{state}^{strength} + (1-\alpha) p(i)_{statistics}^{strength} \quad (1)$$

$V(i)_{strength}$ is the pattern value, where a [0,1] represents the proportion of $p(i)_{state}^{strength}$ and $p(i)_{statistics}^{strength}$. When a = 1, it means only considering the probability that the combination will be followed during the current game. $p(i)_{statistics}^{strength}$ is calculated by probability knowledge. For example, when a player hands red jack, he only needs to calculate the probability that the opponent may have a bomb.

The total pattern value is the sum of all combination cards, and the calculation formula is shown in the formula:

$$V_{strength} = \sum_{i}^{n} (\alpha p(i)_{state}^{strength} + (1-\alpha) p(i)_{statistics}^{strength}) \quad (2)$$

n in the formula is the total number of all combination cards. $V_{strength}$ can not only reflect the strength of the hand, but also guide the arrangement of the cards, judge what cards can be played and if what cards are played, it will cause the hand to become very poor, and basically can't win the game.

E. DQN Architecture

Our DQN model uses the Tensorflow to build the network. Before the data is inputted into the DQN model, this model needs to be set the parameters to build the network, and then the model could be trained. Because the model is prepared to train for DouDiZhu, and this game has many different card combinations, and an agent can play any combination in the game. The input node numbers are the numbers of all card combinations. For the hidden layer 1, it has 256 nodes (neurons) with Relu activation function. For the hidden layer 2, we also set 256 nodes (neurons) with Relu activation function. For the output layer, there are the same numbers of inputs. These layers are fully connected. For the parameters weight and bias, we use the tf.random-normal-initializer(0, 0.3) to initialize the weight value, and tf.constant-initializer(0.1) to initialize the bias value.

The reward is essential to influence the agent to select the action. In most cases, the agent has many different card combinations to select. So, we set different reward values for different card combinations (actions), like the model gets a minus reward if the agent can play a card or cards but does not play any card. The other reward rules are shown below:

- The model gets a minus reward when it selects Bomb or Rocket if it also has other selection of card combinations.
- The model prefers Single Sequence cards, more rewards with the longer sequence.
- The model prefers Double Sequence cards, more rewards with the longer sequence.
- The model prefers Triple with Single or Pair than Triple solo to get more rewards.
- Win the game, and the model gets rewards; conversely, the model gets minus rewards.

III. Implement result and analysis

A. Implement

We developed DouDiZhu game On Cocos2d-x 4.0 and easily packaged our program by Enigma Virtual Box into exe with its image resources and dll files.

You only need to click the correct buttons and can start game, exit game. Besides, you can choose cards you want to play or cancel chosen cards by left-click or right-click.

B. Experiment results

After analysing the result of three models, DQN performed best, followed by the Q-learning model and the Zhou rule-based model. Comparing with the random selection method, the winning rate of DQN is higher than others 10 percent. And competing with human players, DQN has 47 percent chance to win on average, which is similar to ordinary people.

IV. Conclusion

Our DouDiZhu game system combines fun and challenging. With the help of the deep learning algorithm, the difficulty of human-machine combat will be increased. The entire system is easy to deploy and can develop games for different platforms in the future. And we are going to analyze the implicit information of the handed cards from player's teammate and do a research owing to the information exchange between AI players.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] Pomerol J C. Artificial Intelligence and Human Decision Making[J]. European Journal of Operational Research, 1997, 99(1): 3-25.

[4] Sumari A, Ahmad A S, Wuryandari A I, Sembiring J. An Introduction to Knowledge-Growing System: a Review on the Novel Field on Artificial Intelligence[C]//In Proceedings of KNASTIK, 2013: 55-67.

[5] Ingrand F, Ghallab M. Robotics and Artificial Intelligence: a Perspetive on Delliberation Function[J]. AI Communication, 2013, 27(1): 63 -80.

[6] Yolanda G, Greaves M, Hendler J, Hirsh H. Amplify Scientific Discovery with Artificial Intelligence[J]. Science, 2014, 346(6206): 171-172.

[7] Rhalibi A, Wong K W. Artificial Intelligence for Computer Games: An Introduction[J]. International Journal of Computer Games Technology. 2009, 12(3): 351-369.

[8] Greif A. Economic History and Game Theory[J]. Handbook of Game Theory with Economic Applications, 2002, 5(3): 1989-2024.

[9] Billings D, Davidson A, Schaeffer J, Szafron D. The Challenge of Poker[J]. Artificial Intelligence, 2002, 134(1): 201-240.

[10] Bernstein A, Roberts M. Computer v Chess-player[J]. Scientific American, 1958, 198(6): 96-105.

[11] Billings D, Burch N, Davidson A, et al. Approximating Game-Theoretic Optimal Strategies for Full-scale Poker[C]//International Joint Conference on Artificial Intelligence. 2003: 661-668.

[12] Billings D, Kan M. A Tool for the Direct Assessment of Poker Decisions[J]. International Computer Games Association Journal, 2006, 29(3): 119-142.

[13] Berliner H. Backgammon Computer Program Beats World Champion[J]. Artificial Intelligence. 1980, 14(2): 205-220.

[14] Campbell M, Joseph H, Feng H. Deep Blue[J]. Artificial Intelligence, 2002, 134(1): 57-83.

[15] Rubin J, Watson I. Computer poker: A review[J]. Artificial Intelligence, 2011, 175(5): 958-987.

[16] Papp D. Dealing with Imperfect Information in Poker[M]. University of Alberta, 1999, 12(24): 212-230.

[17] Johanson M B. Robust Strategies and Counter-strategies: Building a Champion Level Computer Poker Player[C]//Masters Abstracts International. 2007: 57-65.

[18] Heinrich J, Silver D. Self-play Monte-Carlo Tree Search in Computer Poker[C]//In Proceedings of the American Association for Artificial Intelligence Workshop on Computer Poker and Perfect Information, 2014: 19-25.

[19] Billings D, Davidson A, Schauenberg T, et al. Game-tree Search with Adaptation in Stochastic Imperfect-Information Games[M]. Springer Berlin Heidelberg, Computers and Games, 2006: 21-34.

[20] Ponsen M, Ramon J, Croonenborghs T, et al. Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker[C]//In Proceedings of American Association for Artificial Intelligence, 2008: 1485-1486.

[21] Teófilo L F, Passos N, Reis L P, et al. Adapting Strategies to Opponent Models in Incomplete Information Games: a Reinforcement Learning Approach for Poker[M]. Autonomous and Intelligent Systems, Springer Berlin Heidelberg, 2012: 220-227.

[22] Ekmekci O, Sirin V. Learning Strategies for Opponent Modeling in Poker[C]//In Proceedings of Workshops at the 27th American Association for Artificial Intelligence Conference on Artificial Intelligence, 2013: 6-12.

[23] Heiberg A. Using Bayesian Networks to Model a Poker Player[C]// Workshops at the 27th American Association for Artificial Intelligence Conference on Artificial Intelligence, 2013: 30-34.

[24] Zinkevich M, Johanson M, Bowling M, et al. Regret Minimization in Games with Incomplete Information[C]//Advances in Neural Information Processing Systems. 2007: 1729-1736.

[25] Risk N A. Using Counterfactual Regret Minimization to Create a Competitive Multiplayer Poker Agent[D]. Mastet's Thesis, University of Alberta, 2010: 159-166.