Dou Di Zhu With AI Methods

CS297 Report

Presented to

Professor Chris Pollet

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS 297

By

Xuesong Luo

December 2019

Abstract

My purpose is to design and create a better AI model than others to play the Dou Di Zhu. And I finished four deliveries to prepare the basic AI model. Dou Di Zhu is a card game that is famous in China, and people like playing it with smartphones online. Sometimes, people need to use an AI to help them to play the game when they need to take a phone call or something else. There is some existing research on training different AI models to play Dou Di Zhu, like the Rule-Based, Decision Tree, and Q-Learning algorithm. This semester, I prepared four delivers as pre-operation for the next semester to find the best AI model for Dou Di Zhu.

TABLE OF CONTENTS

I. Introduction

The first playing cards appeared in the 9th century during Tang-dynasty China, and the first reference to the card game in world history dates no later than the 9th century. Right now, like Blackjack, Texas Hold'em these famous card games, are also famous for gambling. Dou Di Zhu is a card game that is famous in China and people also like to play it with gambling. My project is about to design and create the best AI model for playing the Dou Di Zhu.

Dou Di Zhu needs three players, two players are one side called "peasants", and another player is another side called "landlord", each "peasant" has 17 hand cards, and "landlord" has 20 hand cards. If one of the "peasants" plays all the hand cards, the "peasants" side wins the game, then the "landlord" needs to lose the scores or money to the two "peasants". Otherwise, the two "peasants" need to lose the scores or money to the "landlord".

In China, people like playing Dou Di Zhu online on their smartphones. When people play a game on the smartphone, sometimes, they need to face a special situation to stop or quit the game, such as a phone call come inside, they have to answer it. In this circumstance, players need a smart Dou Di Zhu AI to handle the special situations to avoid losing the game. Most of Dou Di Zhu online games include the gambling content, for example, every player has some basic scores at the beginning, and the players can make more scores by winning the game and lose scores by losing the game. If their scores are zero, they cannot play the game and need to wait for their basic scores automatic recovery (like recover 1 score per hour) or spend money to buy the scores.

Some research papers presenting different methods to train the AI for playing Dou Di Zhu, like the Rule-Based, Decision Tree, and Q-Learning algorithm. There is one paper, uses the Rule-Based method, it sets many different rules to simulate the human behaviors for playing Dou Di Zhu. Another paper is about Decision Tree, it is similar to the Rule-Based model, it picks the best action beyond the previous actions. The newest paper uses the Combination Q-Learning algorithm. This model has 2 steps: decomposition stage and the action stage. In decomposition stage, the model decomposes the hand cards first, and based on the decomposed card, the model finds the best cards to play in action stage.

In this paper, the section two is talking about the application for multiple players' Dou Di Zhu, how to create a basic Dou Di Zhu game that can be played by multiple humans. The section three is talking about the Black jack with Q-learning, it's used to learn how to design and create a Q-learning algorithm on simple card game Black jack. The section four is about Dou Di Zhu with Q-learning, and the section five is talking about re-implement Rule-Based for Dou Di Zhu.

II. Application for Multiple Players Dou Di Zhu

My first deliverable was to implement an application that for multiple players' Dou Di Zhu, this application is the basic game that my future AI model can use it for training and testing. This web online Dou Di Zhu game that supports three players can play with others at the same time. I used the JavaScript to code the server and client and Socket.io to connect the server and client. On the server-side, use the Node.js, at the client-side, use the React.

For this application, it needs three players to start the game. After each player clicks the "begin" button, the game will start, and each player will get their shuffle hand cards. The deck will be shuffle by using the Fish-Yate random algorithm and then separate and deal for three players. If one player hopes to become the landlord, he/she can click the "landlord" button to become the landlord and get three more hand cards, of course, the other two players become the peasants at the same time.

When a player plays cards, the player action will be check by some steps before the cards can successfully play. Firstly, the player would be checked if it is his turn. If yes, the next step is to figure out these cards type and compare them with previous cards that the last player plays. After the player succeeds to play the cards, the server will check the player's reminder hand cards. When the number of player hand cards is zero, it means the game is over and the system will find the player's partner at the same time if the player has. So if the player is landlord, there is a toast that show him or her the game result: win; and the other two peasant players will show a toast: lose. If the player is peasant, this player and the other peasant player will show a toast: win; and the landlord player will show a toast: lose.

III. Black jack with Q-learning

My second deliverable was to implement a Q-learning algorithm for Blackjack. I use Python 3 to code this part. Compared with the Dou Di Zhu, Blackjack is a simple card game. The Q-learning algorithm for Blackjack can teach me the Q-learning logic for the card games and how to code a Q-learning for card games. I coded this preparation for the Q-learning algorithm for Dou Di Zhu which will be used in my project.

In my Blackjack game, there are two sides: a player and a dealer. At the beginning of the game, the player and the dealer both get two cards and the dealer will show off one card to the player. Then the player could decide to hit more cards or stop to finish his/her turn. My Q-learning algorithm is based on the player, to analyze the player hit card or not. When at the dealer turn, it will follow the rules: if the dealer's card score is small than 17, the dealer should keep hit the card. After the player's turn and dealer's turn are both over, base on the Blackjack rules, compare their scores to confirm the winner.

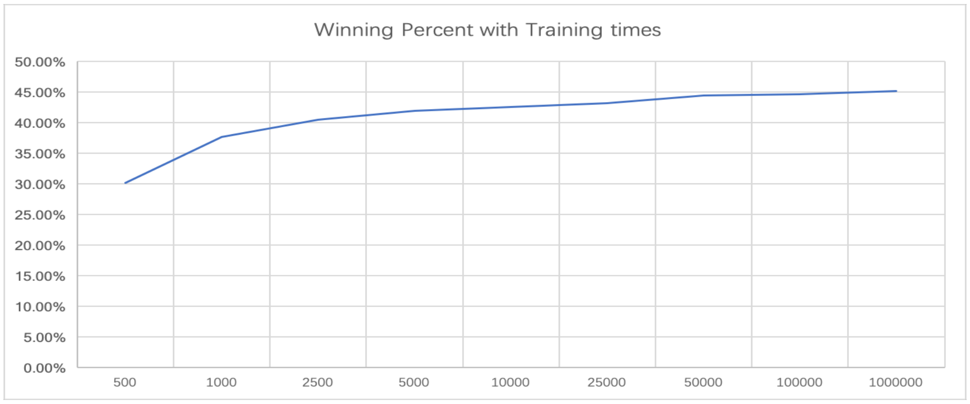
I prepare four shuffled decks for this game. Each time the system randomly chooses a deck for dealing cards. If the number of cards in any one deck is less than 30, this deck will be reset and reshuffled.

At the Q-learning strategy of the player, I set a dictionary as the matrix. The key is player's current score and dealer's one showing card score, and the values are winning times. If the number of values is bigger than zero, it means that hit a card has a higher probability than stop the turn to win the game, if the number of values is small than zero, vice versa. At the beginning of the training, the matrix is empty, so the agent will random to select hit a card or stop. After this turn is finished, the agent will update the win or lose time for the key of the current score. If the agent of player wins the game, the key's value will add one, if lose the game, key's value minus one. After multiple times of training, the player agent will get a huge dictionary matrix. Base on this dictionary matrix, in the test, the player agent will know when to hit a card or stop this turn.

The proportion of training times and testing times is 5:1. I tried nine different training and testing times to check this Q-learning algorithm. With the training times increase, the winning rate also increases.

Training times	Testing times	Wining rate
500	100	30.21%
1000	200	27.70%
2500	500	40.52%
5000	1000	41.90%
10000	2000	42.59%
25000	5000	43.19%
50000	10000	44.44%
100000	20000	44.66%
1000000	200000	45.17%

Transfer these data into a graph as below:



IV. Dou Di Zhu with Q-learning

My third deliverable was to implement a Dou Di Zhu with computer player. I prepared three AI agents with separate Q-learning matrices. This is the kernel part of my project, for my next semester, I will further research and optimize the Dou Di Zhu AI model based on the current Q-learning model.

The agents will follow the order to play the cards. When an agent needs to play the cards, the first thing is the hand cards decomposition. I code some functions to figure out all different types of cards and then decompose the hand cards by some steps.

In the decomposition hand function, the current player hand cards should be input into the function. Then, the function checks the possible card type step by step. If one of the card types resist, the cards belong to the card type need to save separately and remove from the hand cards. The rest of the hand cards will be checked by the next card type function until the function goes over all card type functions.

After finish the decomposition cards, there is another function to find out the agent all possible cards could play which based on Dou Di Zhu game rules, and save these cards independently.

The Q-learning strategy is that every player has themselves Q-learning strategy matrix. At each round, agents save themselves played cards with previously played cards as the combined key into a temp list. After this round of the game is over, all combined keys saved in the temp list should update their winning time or losing time in the matrix.

This simple Q-learning algorithm works not very well on Dou Di Zhu game, it only can teach these agents playing Dou Di Zhu like new players. My training record as follows:

1000 training times, 200 testing time result:

```
    Player1 Winning Percent: 65.0%
    Player1 Peasant Winning Percent: 68.32%
    Player1 Landlord Winning Percent: 51.28%
    Player2 Winning Percent: 52.0%
    Player2 Peasant Winning Percent: 63.64%
    Player2 Landlord Winning Percent: 34.18%
    Player3 Winning Percent: 47.5%
    Player3 Peasant Winning Percent: 60.17%
    Player3 Landlord Winning Percent: 29.27%
```

10000 training times, 2000 testing time result:

Player1 Winning Percent: 64.35%
Player1 Peasant Winning Percent: 65.97%
Player1 Landlord Winning Percent: 59.11%
Player2 Winning Percent: 53.65%
Player2 Peasant Winning Percent: 60.15%
Player2 Landlord Winning Percent: 40.15%
Player3 Winning Percent: 42.05%
Player3 Peasant Winning Percent: 51.87%
Player3 Landlord Winning Percent: 29.5%

The conclusion shows me that, the simple Q-learning cannot play the Dou Di Zhu very well. Between the 1000 times training and 10000 times training, the testing result does not have many differences.

V. Re-implement Rule-Based with Dou Di Zhu

My last deliverable was to re-implement an existed AI model Rule-Based system for Dou Di Zhu. After I finish my Q-learning algorithm, I can compare myself AI model result with other existed AI model results.

This Rule-Based system has two strategies for agent play cards: one is agent playing cards directly strategy; the other one is agent following cards strategy. In the playing cards directly strategy, the agent will follow the below order to select the card combination: *Triple with x, Sequence, Pair, Single, Bomb, Rocket*. After confirming the card combination, select the minimum cards from this card combination to play directly. In the following cards strategy, based on the previous player's cards combination, find the minimum cards that can beat them.

Because it's based on Rule-Based algorithm, it doesn't need training. After testing for 1000 times:

Player0 peasant winning rate: 56.14%
Player0 landlord winning rate: 41.87%
Player0 winning rate: 51.4%
Player1 peasant winning rate: 57.98%
Player1 landlord winning rate: 45.54%
Player1 winning rate: 53.8%
Player2 peasant winning rate: 56.29%
Player2 landlord winning rate: 42.17%
Player2 winning rate: 51.6%

Each player has a similar winning rate, and the result is stable. I run it many times, and the results do not have many differences.

Conclusion

This semester, I prepared the Dou Di Zhu multiple players game application, and through coding the Black jack with Q-learning and Dou Di Zhu with Q-learning step by step, I already prepared the project basic part of the project. At next semester, I will try to figure out how to combine the Neural Networks with Q-learning algorithm to upgrade my AI model for the Dou Di Zhu to create a best AI model for Dou Di Zhu.