AI for One-card -Korean card game-

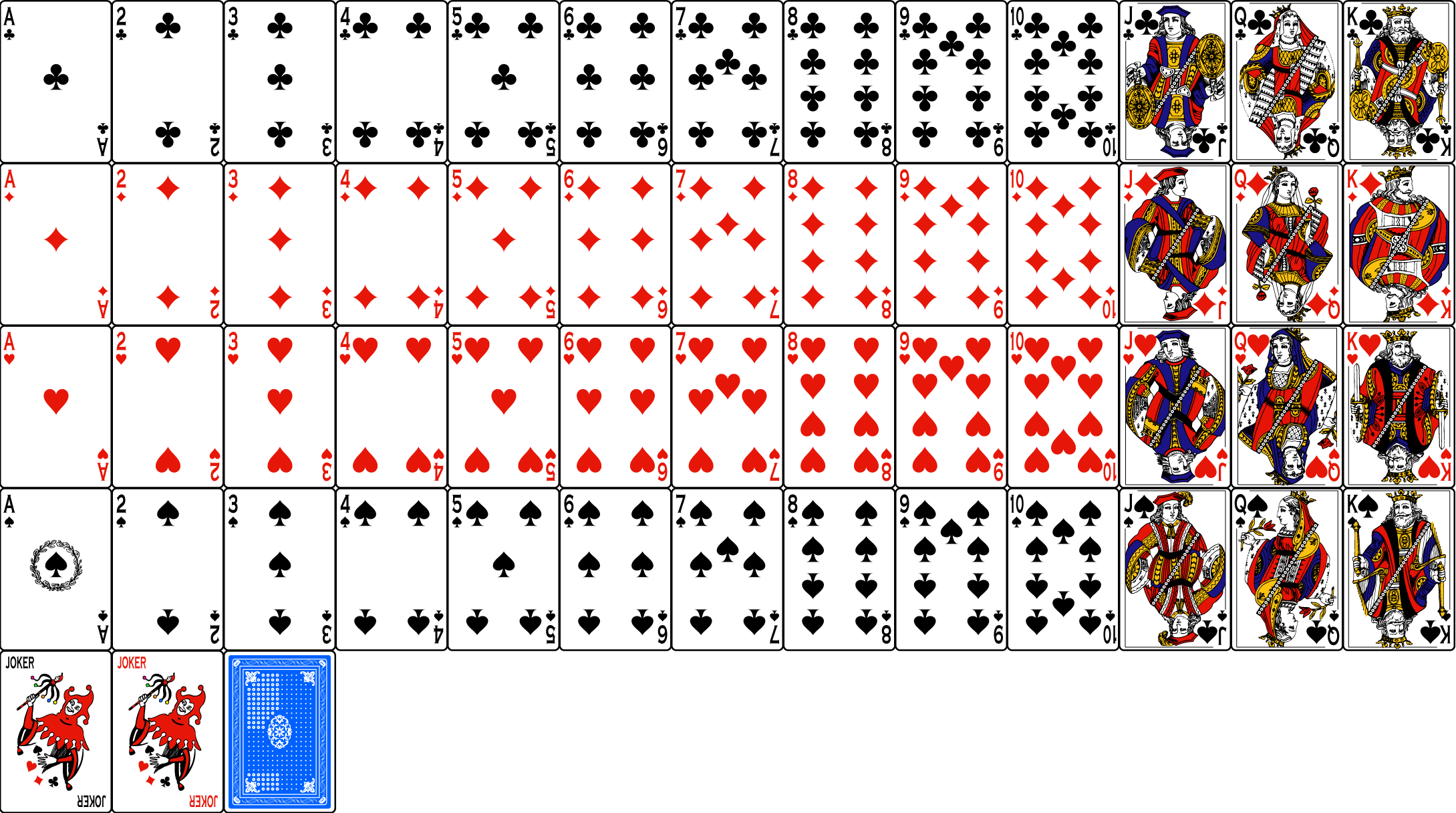
Sung-gon, Ha

Abstract

This report is about developing One-card AI and improving its performance. Since One-card is nondeterministic and partially observable, which means opponent’s cards are unknown, I will use expectiminimax algorithm with stochastic game tree. So, game tree involves chance node. So, I made algorithm and implement as a python code in a form that people can play. So, if a player run the code, then a player can play One-card game with AI. Since One-card is a stochastic game, AI can't have win rate of 100%. But it has win rate about 56%. It is definitely higher than 50% which is win rate of randomly choosing.

Introduction

AI is being used in many fields. Natural language processing, Decision support system, Machine learning, and also Games. In games, there are already famous AIs. Alphago, a baduk AI and Deep blue, a chess AI. Also, PC game StarCraft AI is currently under development. There are many games, but I will focus on card game, which called One-card. One-card is a shedding-type card game. It is played with an ordinary poker deck and the objective is for a player to empty their own hand while preventing other players from emptying theirs. It is not exactly known about its origin, but commonly played in South Korea.



Now, We should know about the rules. To play One-card, there are must 2 or more players and standard 54 playing cards including jokers. But in this report, I assume that there are 2 players. At the beginning of the game, the dealer deals out 7 cards for players. Each players must hide their cards from other player. Then one card from the top of the deck is laid face-up in the middle of the table to form the discard pile, and the rest of the deck laid face-down beside it to form the stock pile. If stock pile gets exhausted, a player takes all but the top card from the discard pile, shuffle them, to form a new stock of cards. The player to the dealer's right plays first. But in this study, player plays first. The player places a card of the same shape or the same number at the top of the discard card pile from his hand. If players don’t have such card in his hand, they must draw a card from the deck. The players then take turns placing or drawing cards and the first player who plays all his or her cards out wins the game. This is the most basic rule. There are many variations on the one card, but this rule remains unchanged.

Basic rules are so simple, so for more fun there are action cards and attack cards. These special cards can be changed since there are many variations. So, before the start of a game, players should decide the rules.

The rule used in this study is the most common rule with little change. There are 3 attack cards 2, A, Jokers. When an attack card is played, the next player must draw one or more cards, or play another attack card. If another attack card is played, damage adds up (2♠ followed by A♠ would have a total damage of 7). In most games, K, Q, J, 7 have special ability, but in this study, they don’t have any special ability. 2 damages level of 2 and other player can play only different shape of 2 or same shape of A or Jokers. A damages level of 3, especially A of spades damages level of 5 and other player can play only different shape of A or Jokers. Black Joker damages level of 7 and other player can play only colored Joker. Colored joker damages level of 10 and cannot be countered. There is also a bust rule. When a player has more than 20 cards, then the player lose. When I played a one-card game with some friends of mine, I found very strange situation. That was one turn kill. Consider this situation, let a player has black and colored jokers and AI has more than 3 cards. First a player plays a black joker. then the AI can do anything except drawing cards. Then AI has more than 10 cards. Next, a player plays a colored joker. Then AI must draw 10 cards. Since AI has more than 20 cards, a player wins. It is very interesting situation, So I will apply bust rule in my model. If player has more than 10 cards, and AI has colored joker, even if attack card is not playing, AI plays colored joker. If player has more than 13 cards, AI has black joker, and colored joker is in AI’s hand or stock pile, then AI plays black joker. In both case, AI must win.

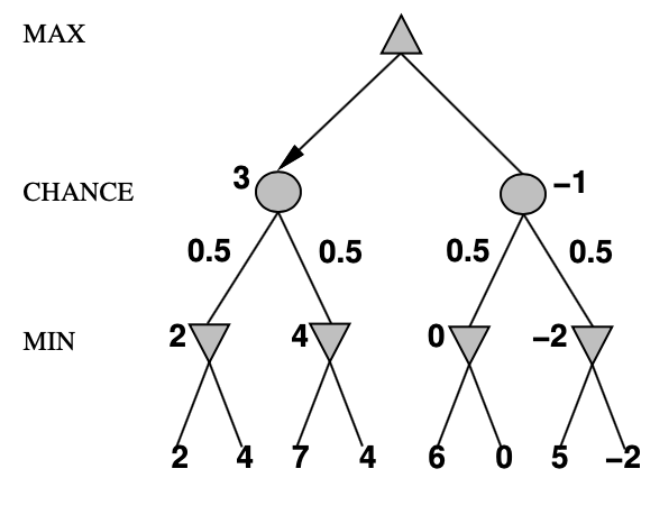
What I want to do is developing the One-card AI for a single player with AI has high win rate. So, the harder to win against the AI, the better. Since One-card is only played in South Korea, One-card AI is very unique and by applying this algorithm, I can develop other shedding-type card game AI easier. Actually One-card AI is already exists. On Google play store, there are many one-card applications. What is new compared to these is AI’s performance. Other AI just play their cards randomly. But I will try AI compute the probability and do the best. So, I applied the minimax algorithm I learned in class. One card is a probabilistic game, so it is difficult to use a normal minimax. Therefore, I used expectiminimax, which utilizes the chance node. Therefore, AI calculates with probability which cards are in my hand and multiplies the probability and the value -obtained by utility function- and plays so that AI has high gain and player has less. When AI attack, it is better to play in order from the weaker than the Joker from the beginning. Utility function which I named value function, makes AI plays attack card more efficiently. If attack card 2 is playing and AI has 2, A, black joker, colored joker. Even though AI has both 2, A, and joker, it is not good to play joker first. The most effective attack method for one card is to play with the lowest damage card among the attack cards you have that can respond to the last attack card. So, if attack card is playing, AI plays attack card with the lowest damage in AI’s hand.

The code to play One-card already exists. Therefore, my goal is to use this to create an AI that acts according to an algorithm. The final goal is for AI to have a win rate of over 50%. If two players playing a game by randomly selecting one card from the playable cards in their hands, each win rate will be close to 50%. Therefore, if the AI has a win rate of 50% or more, it can be said that it uses a better strategy algorithm than randomly choose.

One-card is partially observable since each other doesn’t know opponent’s cards. Also, it is non-deterministic because when the game start, deck is shuffled so player’s cards are randomly picked each time. I learned non-deterministic game model using chance node and game theory and I will use expectiminimax algorithm in this study. It is a minimax for imperfect information. Utility function is also important. I make value function. It acts like a utility function. Value is given to each card differently depending on whether the attack card is playing or not. When attack card is playing, Since AI must play in order from the attack card with the lowest damage, the value is given in order so that the card with the lowest damage given the lowest value and the card with the highest damage given the highest value. If attack card is not playing, Since the AI must collect attack cards and discard the useless cards first, the value of the attack card is low and the value of other cards is high. And since the bust rule is applied, if the player can bust the card played by the AI, a high value is given without reference to attack card is playing or not.

I can learn how to improve game AI for better performance how to create appropriate utility function for given situation. There are many games that hard to play than One-card like Poker or Hanabi. Developing the One-card AI can helps doing my next challenge, making more complex game AI.

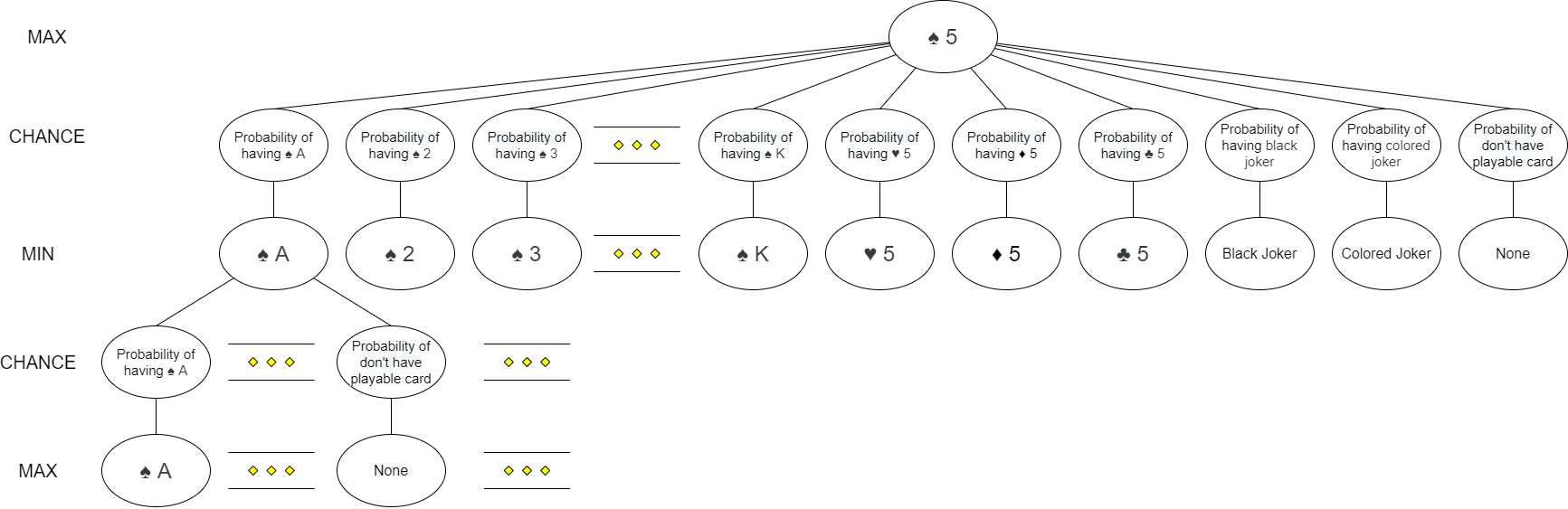
The one-card is nondeterministic and partially observable, so it is difficult to use the normal minimax algorithm. So, to make One-card AI, I used expectiminimax algorithm. It is different from the normal minimax. It has a chance node.



The figure is an example of expectiminimax. The values between chance node and min node represent the probability for each node. In this case, it means that the probability of 2 on the far left of the min node is 0.5, and the probability of 4 on the right is 0.5. The value of the chance node is the product of the utility of the min node connected below and the probability for that node. At the top max node, the value of the chance node with the highest value is selected.

Method

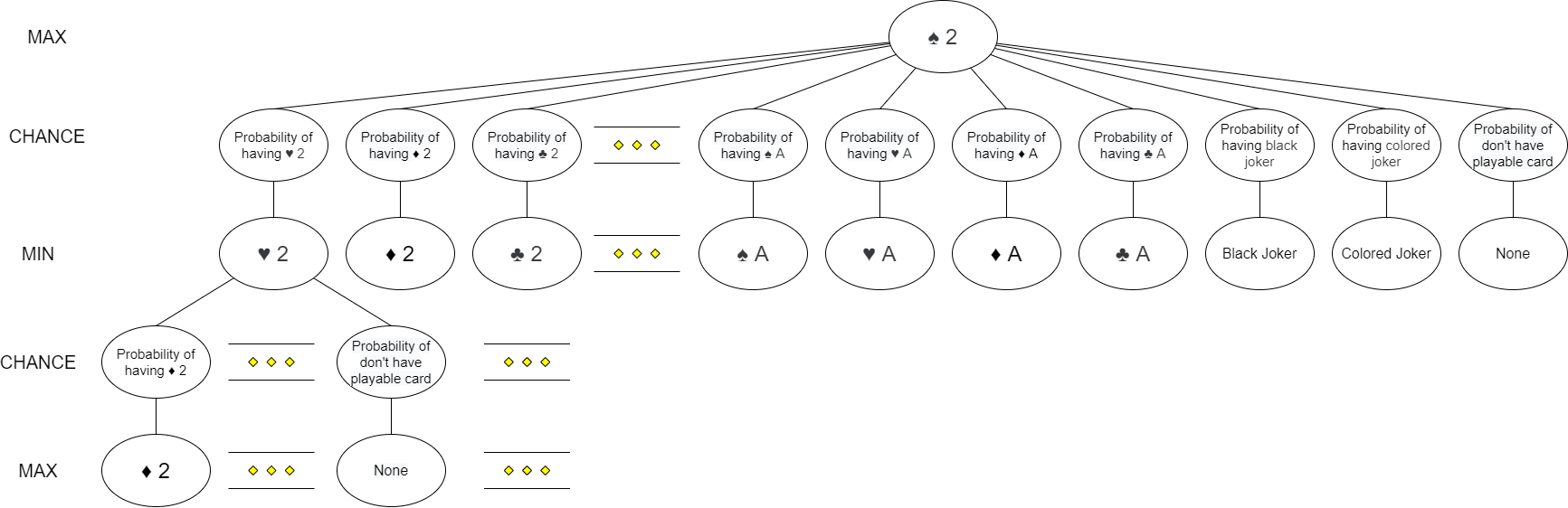
Based on the expectiminimax algorithm, Algorithms take different forms for different cases. First of all, it is divided into two types, if the attack card is played or not, and if the attack card is played, there are 3 cases, 2 is played, a is played, and black joker is played. The algorithm when attack card is not played is as follows.



The top node is the current AI card. For all the cards the AI has, calculate the value using the algorithm as shown in the figure, and play the card with the largest value. The figure assumes the case for ♠5. The chance node in the second row is divided into the cases of cards that can be played for ♠5 and the cases where there are no cards that can be played. The cards that can be played for ♠5 are ♠A, ♠2, ♠3, ..., ♠10, ♠J, ♠Q, ♠K, ♥5, ♦5, ♣5, black joker, colored joker. The node on the far right is when there are no cards to play. The min node connected to each chance node has a min value. Therefore, the chance node is the product of the probability that the player has each card and the min value. In particular, there is no chance node under the none node, and a value of 1 is given to the none node. The probability that a player has each card is expressed in combination. If the number of card in the current player's hand is a and the number of cards in the deck is b, the number of cases of player`s cards is and the number of cases that a player has a particular card is . If the number of playable cards in the stock pile or AI's hand is c and number of playable card is d, the number of cases that player doesn’t having a playable card is . However, if the card is in AI’s hand or in the stock pile, the probability is zero. For example, if there are 30 cards in the deck and the player has 5 cards and ♠ A is not in AI’s hand and stock pile, the probability that the player has ♠ A is = = 14.29%. If 2 of playable cards are in stock pile, the probability that the player doesn’t have playable card is = = 3.58%. The chance node under the min node is the probability that the AI has that card. The AI can see his card, but in this case it's the player's turn, so player doesn't know what card the AI has. Therefore, player will calculate the card that the AI will have and play the card with the lowest AI gain. The max node connected below represents the value of each card. At this time, the cards are assumed to be owned by the AI, so the value is determined by how much the AI can benefit from playing this card. This algorithm is used only when the card is not a joker and returns 0 when the card is a joker. The reason is to prevent the AI from playing the joker when the attack card is not playing and there are other cards that can be played besides the joker.

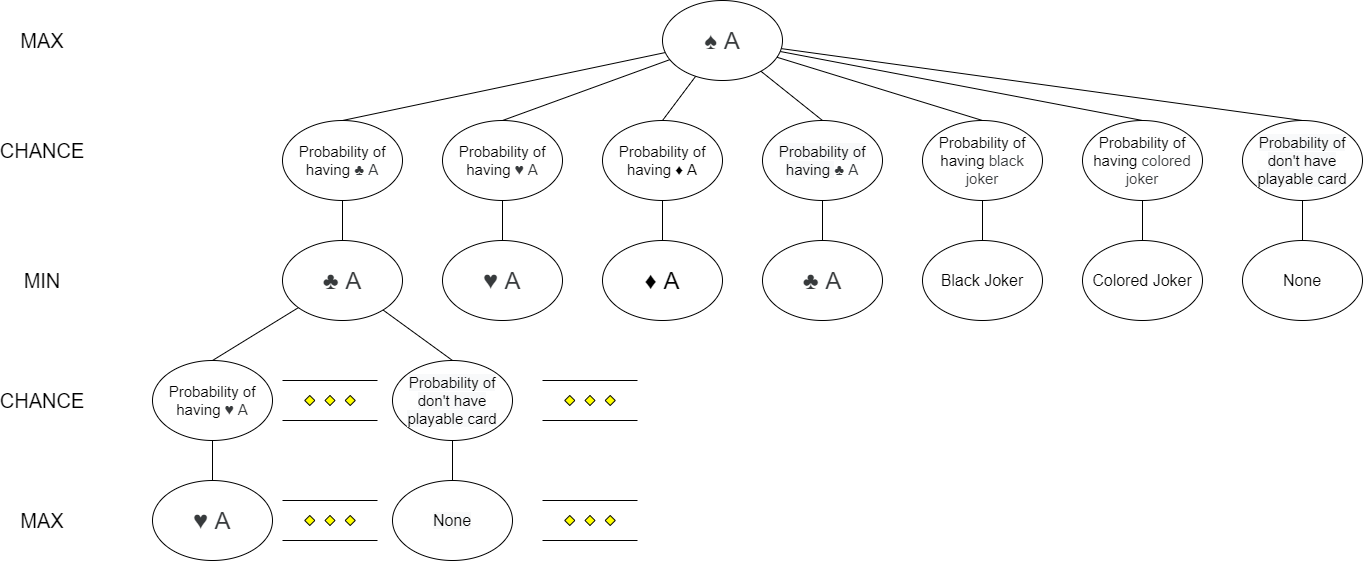
To sum up, determine the value of each card from the max node at the bottom, multiply the probability that AI has that card, and enter it into the chance node in the fourth row. In the third line, select the smallest value among the chance nodes associated with each min node and enter it in the min node. Multiply the card in each min node by the probability that the player has that card and enter it in the chance node in the second row. Finally, select the highest value among the chance nodes connected to the top max node and enter it in the max node. This is repeated for all cards in AI’s hand, and the card with the highest value is played.

The algorithm when attack card 2 is played is as follows.



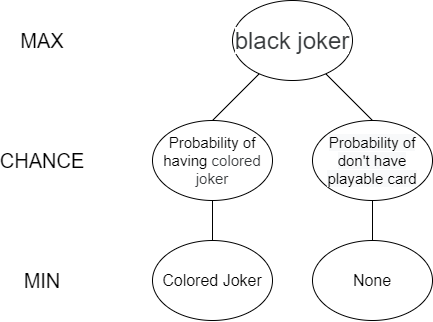
This is the same as when the attack card has not been played, except that the type of card in the min node has changed. The figure assumes the case for ♠2. The cards that can be played for ♠2 are ♥2, ♦2, ♣2, ♠A, ♥A, ♦A, ♣A, black joker, colored joker. The method of determining the value of max node is as above.

The algorithm when attack card A is played is as follows.



The figure assumes the case for ♠A. The cards that can be played for ♠A are ♥A, ♦A, ♣A, black joker, colored joker.

The algorithm when attack card black joker is played is as follows.



In this case, there is no chance node below the min node. Because there are no cards to play against colored joker. For the same reason, there is no algorithm for colored jokers. In that case, there is no card that can be played, so the value is not calculated, and just draw cards.

Now we need to determine the value function. Basically, the value function was set so that the higher the benefit to AI when playing a card, the higher value given to the card. The value function was calculated by dividing the case when the attack card is being played or not. If the attack card is played, the value of 2 is 6, the non-spade A is 5, the spade A is 4, the black joker is 3, the colored joker is 2, and 1 for otherwise. The higher the damage is, the lower the value is assigned, so that the attack card with the lowest damage in AI's hand is played first. If the attack card is not played, 2 is -1, non-spade A is -2, spade A is -3, black joker is -4, and colored joker is -5. For other cards, 100 was basically given, and if there were cards of the same shape or number in the AI's hand, 10 values were added to each card. The reason why the other cards are given a particularly high value is to make AI plays the non-attack card first if the attack card is not playing, and the AI has both attack and non-attack cards. Therefore, if the attack card is not being played, the AI saves the attack card as much as possible and plays the card with the same number or shape on the hand first.

Also, because there is a bust rule, there are cases when AI can win regardless of the player's action. If the colored joker is in a stock pile or AI`s hand, and the player has 13 or more cards, and the AI has a black joker, AI plays the black joker regardless of the value. In this case, AI must win. Similarly, if the player has 10 or more cards, and the AI has a colored joker, AI plays the colored joker. And if the player has 3 or more cards, and AI has colored and black joker, AI plays colored joker. In this case, the player cannot do anything without drawing 10 cards, and the AI plays the black joker immediately. Then AI wins.

Result

The basic frame of code to play one card was made by someone else. What I did was add an AI algorithm to this code so that the AI could behave as an algorithm and add some rules like damage of the attack card, bust rule.

As a result, I made one card AI code that people can play. When you run the code, you can see the top card of the stock pile, the player's hand, the playable cards and the input space. In the input, you can enter the order of cards you want to play from the list of cards that can be played. When you play the card, it goes to the AI’s turn. Then number that how many cards the AI has is showed, and the AI plays the card. Then it becomes the player's turn again. If there is no playable card, it shows how many cards you have to draw and the input space appears. When you press enter, player draws cards and it goes to the AI’s turn. Similarly, if AI doesn’t have playable card, it shows number of cards that AI has drawn and it becomes the player's turn. When the player or AI wins, the winner's name appears and execution ends.

I tried playing the game myself to see if the code works. The original code doesn't show the AI's hand, but I tried to create and play a new code that prints the AI's hand to see if AI works properly. I played it a few times without any conditions, and then I added some attack cards to the player and AI when I started the game. I also gave the AI a joker to play at the start of the game to see if the AI could use the bust rule. In fact, as a result of playing, I have seen that the AI cards are attacked in weak order, and if the AI plays the joker makes the player becomes bust, then the joker is played by AI immediately. When I played a few times, I confirmed that AI won a lot, but for a more accurate judgment, I made a model that calculates the win rate of AI.

I deleted all prints like showing the top card of stock file and player’s hand and changed the code to print only the winner. It is impossible for me to play a lot of times myself, so I make the player randomly picks one of the available card-playable cards-and plays it. At the end of each game, the winner was recorded, and divide the number of AI wins later by the total number of runs to calculate the AI's win rate. As a result of the first 1000 runs, the AI win rate was 55.1%. The win rate is over 50%, so it is a good result. I ran 1000 more times and the AI win rate was 55.15%. Since then, it has been repeated 1000 times, with a winning rate of 56.07% at the 10000th.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trial | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 |
| AI win rate | 55.1 | 55.15 | 54.83 | 55.55 | 56.02 | 56.12 | 55.91 | 55.9 | 55.97 | 56.07 |

Therefore, it can be said that the winning rate of AI is close to 56%. I've achieved the win rate of over 50%, which I initially targeted, so I've confirmed that AI’s algorithm is working well. I also tried the case where both AI and player randomly choose. In this case, 10000 times were executed. In this case, the win rate of AI was 50.32%, which was close to 50%. Considering that One-card is a stochastic game, the more you run, the closer you will get to 50%. As a result, it was confirmed that the algorithm has a higher win rate than randomly choosing. The lower left figure is the result when the AI behaves according to the algorithm and the player randomly chooses 10000 times. The figure on the right is the result of executing 10000 times when both AI and Player randomly choose. The lowest number in the code indicates the AI’s win rate. You can see that when the AI acted according to the Algorithm, it was 56.07%, and when randomly choose, it was 50.32%.

스크린샷이(가) 표시된 사진

자동 생성된 설명 스크린샷이(가) 표시된 사진

자동 생성된 설명

Conclusion

I developed One-card AI for single player. The AI has basic rules and some strategies for good performance. Because there is no research on the winning method yet, it is very unique to study the winning method and even create the AI to which it is applied. I applied expectiminimax algorithm with chance node and implement in python. The expectiminimax algorithm take different forms for different situations. Utility function of algorithm is a value function in my code. The value function assigns different values to the card depending on the situation. The higher the benefit to AI when playing a card, the higher the value of the card. I also using a variety of strategies like AI to starting attack cards with low damage or making player bust. Since One-card game is partially observable and non-deterministic, I expect this AI can’t have a win rate of 100%. However, AI aimed to achieve a win rate of at least 50%, and as a result of playing 10000 games against player who randomly chooses available card, AI achieved the goal with a win rate of about 56%. This algorithm may not be the one that makes AI have the highest win rate. If the value function is modified or other changes are given, the AI may have a higher win rate.

References

[1] Michael van Lent and David Mutchler. A pruning algorithm for imperfect information games. In Proc. of the AAAI Fall Symposium. on Games: Planning and Learning, Raleigh, 1993.

[2] “파이썬 강좌 – 실습 – 원카드 게임 만들기 (1)”, (May 14, 2020), Under The Pencil, https://elvanov.com/1441