Colton Avila

CS531

9 June 2021

Final Project: Gin Rummy AI – Evaluation Function Comparison

Abstract:

Artificial Intelligence programs used to solve games of chance have to consider not only the present and past states, but many of the possible future states with various probabilities. To better understand this process, we modelled an AI that played the classic card game Gin Rummy. Our program aimed to have the lowest hand value in a one vs. one game. It sought to create 3 or more card melds - sequential cards in the same suit or cards with equal rank values – which set the value of the cards equal to zero. To generate the maximum number of melds, our AI implemented a "Myopic Meld" function. This function took the 2-card combinations after drawing from either the discard pile or the deck and measured the number of possible melds that could come from either choice. This evaluation function led us to implement an algorithm where the action that generated the lowest hand value was chosen. We hypothesized that the implementation of this utility evaluation function would improve the performance of the AI compared to an AI that just randomly picking a pile to choose from. The results of our tests cases showed that this "Myopic Meld" function did not produce a better result compared to just randomly choosing a card to pick up. Inclusion of "Opposition Modeling" and "Stubborn Knocking" variables in the evaluation function has been proven to improve performance, but without these extra variables the AI fares worse than random overall.

Introduction:

Gin Rummy is a card game where two players seek to have the minimum hand score at the end of the round. Each player is dealt 10 cards and one card is placed face-up on the discard pile. The cards in hand take their respective ranks as their values (1-10 are worth that number of points) except face cards are assigned the value 10 and aces are assigned the value 1. The values of the cards are added to the total score of the hand; this is called the "deadwood" value. By creating sets of three or more cards where cards are either sequential and in the same suit or the cards are all equal in terms of rank, the players can remove the values of these cards from their deadwood scores. The game is played in turns where each player can draw either from the discard pile, draw from the deck, or knock on the table. If a player has less than 10 cards, they can knock signaling that round will come to an end next turn. The player that knocked must lay down their cards after the other player takes their turn and the other player is allowed to "layoff" any remaining deadwood cards onto the knocking player's hand. If the non-knocking player's hand has a lower value than the knocking player's hand after the layoff's then the non-knocking player wins the round and

receives a 25-point bonus. However, if the knocking player knocks, has a deadwood score of zero, and wins then they receive a 25-point bonus as well.

Much of this paper's information on modelling comes from a recent paper that attempted to evaluate which pile to draw from based on certain utility evaluation functions (Goldman et al. 2021). In this paper they looked at Gin Rummy as a partially observable game where the AI understands what cards might be left in the deck and in the opponent's hand and when they saw them either in the discard pile or in their own hand, they could remove those cards from the set of possible cards they could draw. They created multiple AI configurations that used the previously described "Myopic Meld" (MM) to generate a utility evaluation function, but also included "Opposition Modelling" (OM) and "Stubborn Knocking" variables. OM aimed to pick which card to discard at the end of a turn that had the minimum chance of being picked up by the opponent. This was done by keeping track of all possible melds that the opponent could have and removing melds when their cards were discarded or drawn by the non-opponent player. SK aimed to prevent the opponent from undercutting the player and receiving the 25 bonus points. Goldman's paper created 5 AI models that used different utility evaluation function in their action decisions:

1. MM + OM + SK
2. SK + MM
3. MM + OM
4. SK
5. A "Simple" Model

In this paper we sought to evaluate the performance of the "Myopic Melding" equation in the utility evaluation function without including the "Opposition Modelling" or "Stubborn Knocking" variables. Our proposed model was not evaluated in past work and will be explored below. It should also be noted that in the original paper the "null" or baseline model was not a random-decision based AI, but an AI that drew from the discard pile only if it would generate a meld and drew from the deck all other times.

Methods:

Evaluating Actions

We generated three AI players. The difference is that one had only the Myopic Melding function driving its utility evaluation function, one had the MM function as well as the Stubborn Knocking function, and the last one had no evaluation function and acted randomly. The Myopic Meld Function is listed below:

$$r_m = \sum_{c \in (H_i \setminus W_{H_i})} \frac{|V(c, H_i)|}{|U|} d(c) \quad (1)$$

We show that r_m is the value of the MM function. This value will be subtracted from the deadwood score of each possible hand generated from drawing either the discard pile card or the card from the draw pile. This part of the equation, $c \in (H_i \setminus W_{H_i})$, means that the MM score is generated for every card in a player's hand that is not already in an existing meld. $|V(c, H_i)|$ is the

number of possible melds that can be generated in the future by drawing this card. $|U|$ is the available number of cards left in the deck. $d(c)$ is the deadwood value of the card that is being estimated.

For each action we generate the utility evaluation function value $r_t(h)$ which each player looks to minimize. This value will be assigned to each action: draw the card from the discard pile or draw the card from the deck.

$$r_t(h) = d_{min}(h) - r_m \quad (2)$$

The deadwood score of a player's hand is represented by $d_{min}(h)$ and the player looks to minimize $r_t(h)$ by selecting the action which results in the highest r_m score. For the discard pile the card is drawn and the highest r_m value for any discarded card is set as the $r_t(h)$ value for the "draw from discard pile" action. Drawing from the draw pile averages all possible $r_t(h)$ values by the number of cards remaining in the deck. Which action has the lowest $r_t(h)$ each turn, the AI decides to do that action. For the random AI, it randomly selects one of the piles to draw from but does consider the melds when scoring its hand. This functions a minimax algorithm that only has a depth of 1 layer.

Pseudocode

```
For 100 Cases -> Shuffle Deck, Deal Cards, Set Initial Discard Card:
    If Player hand < 10 & not SK Agent:
        Knock
        LastTurnofGame(); "This lets the other player take more turn before the layoff
                           phase"
    For Players in Game:
        If RandomAgent():
            RandomlyPickCard() & RandomlyDiscardCard()
        Else:
            A = GenerateUtilityofDiscardPileDraw():
                Generate Combinations of All Possible Hands
                Find Myopic Melding Scores For Hands in Combinations
                Subtract MM Score from Deadwood score for each hand
                Find the minimum value of the above value
                Set the utility value of the discard pile above
```

B = GenerateUtilityofDrawPile():

For Each card in the draw pile:

Generate Combinations of All Possible Hands for a card

Find Myopic Melding Scores For Hands in Combinations

Subtract MM Score from Deadwood score for each hand

Find the minimum value of the above value

Set the card's utility value equal to the above value

Average the utility values in the deck and set the "Draw From Deck" action equal to this value

Select the Minimum value between A and B; This tells us which card to draw and which card to discard

Once LastTurnofGame() ends:

Compare scores between players, and calculate winner

Check to see if the non-knocking player can layoff any cards on the knocking players hand

Compare scores between players, and the check to see if the winner changed (this means they should be awarded bonus points)

Add a value to the winner's total, and add a value if the winner won in the layoff phase.

This code does not exist in a database linked to the paper or that we could find, so all code is original.

Results:

To generate our test cases, we pitted two different AIs against each other and compared the number of games won by each. For the first test case we had an AI with only the MM function working to evaluate a state's utility. The other AI was the same, but it knocked only when its deadwood score was zero.

Trial #	Time (s)	AI1 (Myopic Melding)	AI2 (Myopic Melding + Stubborn Knocking)
1	0.01	70 (0)	30 (4)
2	0.02	61 (0)	39 (1)
3	0.03	66 (0)	34 (1)
4	0.01	60 (1)	40 (2)
5	0.02	57 (0)	43 (3)
6	0.02	60 (2)	40 (3)
7	0.02	54 (0)	46 (1)
8	0.02	62 (0)	38 (1)
9	0.02	61 (0)	39 (1)
10	0.03	58 (0)	42 (1)

Table 1. Test Cases for Myopic Melding AI vs. Myopic Melding with Knocking AI.

Looking at Table 1, we found that the MM AI beat the MM + SK AI around 65% of the time. By looking at the values inside the parentheses in the AI columns we can see how many times the AI won during the "layoff" phase of the game where they would gain the bonus 25-points. In a race to 100 points, it is safe to say that waiting to knock at a 0-deadwood score is non-optimal as on average the other player will beat you enough times to outweigh that bonus.

The second test case pitted the MM AI against an AI that chose the draw pile and discard card that is not in a meld at random.

Trial #	Time (s)	A1 (Random Discard & Pickup)	A2 (Myopic Melding)
1	0.02	59 (0)	41 (3)
2	0.02	60 (1)	40 (1)
3	0.02	57 (0)	43 (3)
4	0.03	60 (1)	40 (0)
5	0.02	60(1)	40 (0)
6	0.01	65 (0)	35 (0)
7	0.02	56 (2)	44 (2)
8	0.02	66 (0)	34 (0)
9	0.02	54 (0)	46 (1)
10	0.2	61 (0)	39 (1)

Table 2. Test Cases of Random AI and Myopic Melding AI

Table 2 shows that an AI that plays randomly beats the MM AI approximately 62% of the time. Even though the MM agent performed better than the MM + SK agent, it still could not outperform randomly choosing a card to draw and discard.

Discussion:

It was surprising to see that the MM AI did not beat the random AI. While in the paper the MM + SK model does beat their "Simple" or "null" model and their "Simple" model also had

rules governed its action. Their “Simple” model draws the face-up card only when it can create a meld and draws from the draw pile every other time. It also discards the highest-ranked unmelded card. Our random AI just chooses a pile at random to pick and discards a card that is not currently in a meld at random. It begs the question of just how well we can predict what a winning hand will look like using non-Monte Carlo based AI if we are unable to generate strong evaluation functions. This result also raises the question as why the original paper does not set the “null” AI to having random action but an AI with arbitrary rules. We found that our hypothesis was wrong and that an MM AI cannot beat the random AI.

If this were done again, we would generate all models discussed in the paper and try to determine why the MM function fails at beating the random AI. We would also increase the depth at which the MM function was calculated at. We expect that the deeper the search is using MM the more likely that we would be able to predict actions that will outplay the random AI. We also can modify the probability of drawing cards. In the OM equation, the opponent’s pick-ups from the discard pile are tracked. This means that if the opponent picks up two 2’s then our probability of drawing a two is probably not equal to drawing a 7. It would also be interesting to see how well our random AI compares against all the original paper’s AI models.

Conceptually, I learned a great deal from this experiment. How we model our utility evaluation function for minimax algorithms greatly affects the performance of the AI. Our AI could not outperform a simple random action AI. This leads me to understand that most of the work in developing a strong AI is in the evaluation function and not the actual implementation of the AI. It is possible that the methods in the paper that our work was based on are not very strong when it comes to predicting a “good” action. On the implementation side of things, I learned quite a bit. How to structure my agent, how to establish states in the game, how to evaluate these states, and how to measure performance were all things that I was able to do with the knowledge from this course. For future work, I could see myself implementing a Monte-Carlo AI to try to outperform a better structured minimax AI. I also would really like to see how deep we need to search the minimax tree to get results that definitively beat the random AI.

Bibliography

Goldman, Phoebe, et al. "Evaluating Gin Rummy Hands Using Opponent Modeling and Myopic Meld Distance." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. No. 17. 2021.