Opponent Hand Estimation in the Game of Gin Rummy

Peter E. Francis, Hoang A. Just, Todd W. Neller
Gettysburg College
{franpe02, justho01, tneller}@gettysburg.edu

Abstract

In this article, we describe various approaches to opponent hand estimation in the card game Gin Rummy. We use an application of Bayes' rule, as well as both simple and convolutional neural networks, to recognize patterns in simulated game play and predict the opponent's hand. We also present a new minimal-sized construction for using arrays to pre-populate hand representation images. Finally, we define various metrics for evaluating estimations, and evaluate the strengths of our different estimations at different stages of the game.

Introduction

In this work, we focus on different computational strategies to estimate the opponent's hand in the card game Gin Rummy, i.e. to evolve probabilistic beliefs about an opponent's hidden cards. Hand estimation in Gin Rummy is particularly challenging because of the little information available, the unknown strategy of the opponent, and the stochastic nature of the game.

The origins of the melding card game Gin Rummy (McLeod 2020) are debated: some believe the game stemmed from 19th-century whiskey poker, while others disagree and say the game Conquian makes more sense as its ancestor. Although Gin Rummy was one of the most popular card games of the 1930's and 1940's (Parlett 2008, 2020) and remains one of the most popular standard deck card games (Ranker 2020; BoardGameGeek.com 2020), it has received relatively little Artificial Intelligence (AI) research attention.

We explore variations of previous strategies used on other games, as well as develop some new approaches. These algorithms differ in the way simulated game data was abstracted to make future predictions. The application of Bayes' rule and the use of simple and convolutional neural networks allows us to see how different constructions lead to different estimation powers. In the process of developing a useful image for convolution-based deep learning and pattern recognition, we introduce the idea of an ambi-superpermutation, as well as prove minimal length properties. We also introduce various metrics of evaluating the success of a hand estimation, and show one way this estimation can make a difference in a game play strategy. We conclude with a demonstration of a simple deterministic application of our hand estimation that produces a statistically significant advantage for a player.

Gin Rummy

Gin Rummy is one of the most popular 2-player card games played with a standard (a.k.a. French) 52-card deck. Ranks run from aces low to kings high. The object of the game is to be the first player to score 100 or more points accumulated through the scoring of individual hands.

The play of Gin Rummy, as with other games in the Rummy family, is to collect sets of cards called *melds*. There are two types of melds: "sets" and "runs". A *set* is 3 or 4 cards of the same rank, e.g. $\{3\clubsuit, 3\heartsuit, 3\spadesuit\}$, or $\{K\clubsuit, K\heartsuit, K\spadesuit, K\diamondsuit\}$. A *run* is 3 or more cards of the same suit in sequence, e.g. $\{5\clubsuit, 6\clubsuit, 7\clubsuit\}$, or $\{9\heartsuit, T\heartsuit, J\heartsuit, Q\heartsuit, K\heartsuit\}$. Melds are disjoint, i.e. do not share cards.

Cards not in melds are referred to as *deadwood*. Cards have associated point values with aces being 1 point, face cards being 10 points, and other number cards having points according to their number. *Deadwood points* are the sum of card points from all deadwood cards. Players play so as to reduce their deadwood points.

For each hand of a game, the dealer deals 10 cards to each player. (There are different systems for deciding the next dealer; we will simply start with a random dealer and alternate players as dealer across the entire game.) After the deal, the remaining 32 cards are placed face down to form a draw pile, and the top card is turned face-up next to the draw pile to form the discard pile. The top of the discard pile is called the *upcard*.

In a normal turn, a player draws a card, either the upcard or the top of the draw pile, and then discards a card. The player may not discard a drawn upcard but may discard a card drawn face down. For the first turn, play starts with the non-dealer having the option to take the first turn by drawing the upcard. If the non-dealer declines this option, the dealer is given the option. If both decline, the non-dealer must take the first turn drawing from the draw pile.

In the event that the hand has not ended after a turn with only 2 cards remaining in the draw pile, nothing is scored,

all cards are shuffled, and the hand is replayed with the same dealer.

After a player has discarded, if that player's hand has 10 or fewer deadwood points, that player may *knock*, i.e. end the hand. (Often this is indicated by discarding face-down.) The hand is then scored as follows: The knocking player displays melds and any deadwood. Next, if any deadwood was displayed, the non-knocking player may reduce their deadwood by *laying off* cards, i.e. adding cards to the knocking player's melds. Then, the non-knocking player displays melds and any remaining deadwood.

If the knocking player had no deadwood, they score a 25-point *gin* bonus plus any opponent deadwood points. If the knocking player had less deadwood than their opponent, they score the difference between the two deadwood totals. Otherwise, if the knocking player had greater than or equal to their opponent's deadwood points, the opponent scores a 25-point *undercut* bonus plus the difference between the two deadwood totals.

A player scoring a total of 100 or more points wins the game.

Prior Work

There have been several approaches to estimate the opponent's hand in imperfect information card games. We provide two works that differs completely in their complexity of computation during the game.

- **DeepStack Approach.** DeepStack (Moravčík et al. 2017) was used to solve Heads-Up No-Limit Poker (HUNL) which, similarly to Libratus (Brown and Sandholm 2018), was able to beat professional human poker players. The DeepStack approach is to calculate Counterfactual Regret Minimization (CFR+) to find a mixed strategy that approximates a Nash Equilibrium strategy in order to maximize the expected utility. First, it runs the CFR+ algorithm for a limited moves ahead. Then, later in the game DeepStack uses two different neural networks to approximate the counterfactual value of the hand (one after the flop and one after each turn), which were were trained beforehand, so that it does not to re-run the CFR+ algorithm. Since the counterfactual value of the hand is recalculated after each turn, DeepStack does not need to save the whole strategy throughout the game, and instead performs continual resolving. A similar implementation for Gin Rummy would be too time intensive for our goals, since DeepStack's nature is to consider a new game state and recalculate a new strategy after each turn.
- **AI Factory Approach.** This heuristic opponent hand estimation used in AI Factory Ltd.'s Gin Rummy Free app was described online by Jeff Rollason (Rollason 2007) but not fully specified. Since AI Factory needed to implement a hand estimation for Gin Rummy in their mobile application, it needed to have a fast hand estimator for the AI bot in the game. Their model used the linear evaluation which summed the value of a meld times the probability that this meld can occur in the player's hand, then that sum was subtracted by the value of that meld times the probability that it could happen in the opponent's hand. If the given card can be melded in more sets or runs, then the value is increased, which means that there is a higher chance of getting that specific meld. AI Factory hand estimation also assumed that if the opponent discarded a card or refused to pick up a face up card then all neighbouring cards (cards of the same rank and cards of the same suit with adjacent rank) are less likely to be in opponent's hand. Respectively, if the opponent picked up the face up card, then the neighbouring cards are more likely to be in opponent's hand. Therefore, probability of each card starting with a midpoint probability of 0.5 will be scaled up or down accordingly to the opponent's decision. For example, when an opponent is discarding a $6\heartsuit$, the probabilities of adjacent cards $5\heartsuit$ and $7\heartsuit$ decrease. However, the fact that the opponent might have a meld consisting of $7\heartsuit$ is not taken into account. Thus, the AI Factory approach does not provide flexibility for different player strategies.

Proposed Algorithms

There are multiple ways to approach opponent hand estimation and we show three different models. What is common between the algorithms is their data-driven design: each uses play data from a large sampling of simulated games.

Bayesian Estimation from Abstracted Play Data

One of our novel approaches began as an attempt to take a heuristic approach to opponent hand estimation and derive a similar approach that is Bayesian and play data driven. While the algorithm is not specified in (Rollason 2007), an opponent drawing a face-up card increases the estimated probability of having same rank or adjacent suit cards, and similarly, an opponent discarding a card or refusing a face up card decreases the probability.

Our approach begins with Bayes' Rule and its simple proportional form:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A|B) \propto P(B|A)P(A)$$

Here A represents the atomic sentence that the opponent holds a specific card, and $P(A)$ is the probability that the opponent holds that card. Event B is an observation of an opponent's draw and discard behavior on a single turn. Prior to event B we believe the probability of the opponent holding card a specific card is $P(A)$. After event B, our posterior belief of $P(A|B)$ is proportional to the frequency of observing such behavior of play data where the same draw/discard event while holding that card $P(B|A)$ times our prior $P(A)$.

We could naively collect play data to find the likelihood of event B conditioned on A, but we may apply a helpful abstraction that enforces a rational symmetry concerning (in)equality of the suits of the card in consideration, the card drawn, and the card discarded. For example, we have no reason to believe that the likelihood of the opponent drawing Q♣ face-up and discarding K♠ conditioned on the opponent holding K♣ should be any different than

that of drawing Q♡, discarding K♢, and holding K♡, respectively. What matters is the suited/unsuited relationships of the draw/discard cards to the potentially held card.

For this reason, we abstract the frequency data we collect from play on draw/discard events for each card with respect to:

- Whether or not a card was drawn face-up
- Rank of the face-up card
- Rank of the card discarded
- Whether the card was suited with the face-up card and/or discarded card

Play data was collected from $10,000$ simulated games between two simple Gin Rummy players that (1) only drew face-up when the card completed or extended a meld, (2) discarded randomly from discards that would maximally reduce deadwood, and (3) knocked at earliest opportunity. Far from optimal, this simple play nonetheless was adequate to capture general statistics that adjusted probabilities similar to the AI Factory Ltd. approach. In cases where fewer than 50 abstracted observations were observed, we opted not to update our hand estimation. Cards known to (not) be in an opponent's hand are probability 0 or 1 and are not updated.

Given that each player starts with 10 cards of a 52 card deck, our initial probability for a player holding a card is 0 for those cards in our own hand, and $\frac{10}{42}$ for the 42 cards that could be in the opponent's 10-card hand. After each opponent turn, we checked our frequency data to see if we had a minimum of 50 observations from which to form our expectation. If not, we conservatively do not revise our hand estimation. If so, we multiply each unknown card estimate by out frequency-based likelihood.

We then renormalize probabilistic estimates of unknown cards so that the estimates sum to the number of unknown cards in the opponent's hand, and then fit the probabilities through a logistic curve to ensure they are no more than 1.

Simple Neural Network

In this approach, we trained a multi-layer network on a few features of the current game state to predict the opponent's hand. Figure 1 shows a diagram of the input array to the network (OHE denotes "one-hot encoding" over the range of indices).

The layers of the network are all Dense and are activated with the sigmoid function. Each has size 100, 85, 60, and 52 (this layer has one node for each card), respectively. Batch sizes and epochs were hand-tuned.

Since this network is trained on the current probability distribution, multiple cycles (generations) of data-collection and training were used to arrive at a final model.

Shankar CNN

This last model is largely influenced by Dr. Pramod Shankar's method of "Figuring the Opponent's Hand" and how he determines safe discards using a *plug rule* (Shankar 1994). First Shankar's Plug Rule is used to create a 4×13 array. Then, the rows of the array are duplicated, permuted, and ordered to create a 17×13 array using the (newly coined) ambi-superpermutation on four elements. Finally, the 17×13 array was processed through the Convolutional Neural Network with ReLU activation functions, filter sizes 16, 32, and 64, each with kernel size 3×3, and max pooling with size 2×2.

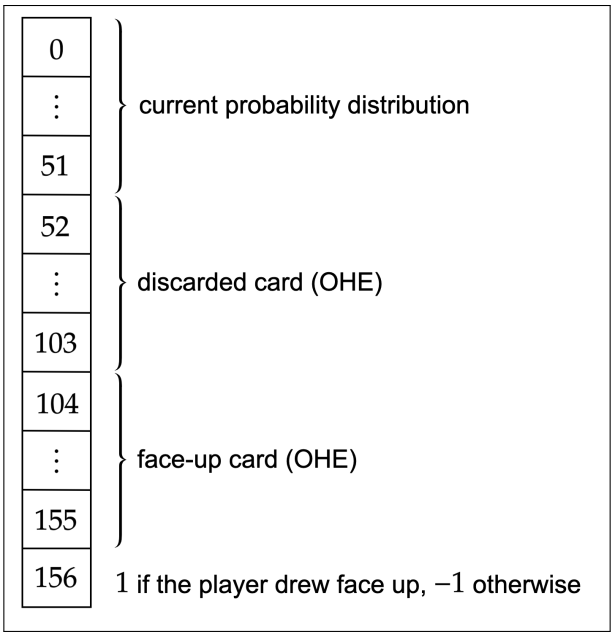

Figure 1: Array Input to Simple Neural Network

The Plug Rule The Plug Rule is used to inform the player about which cards are safe discards and which cards are in the opponent's melds. We applied Shankar's Plug Rule to our hand estimator, since it assures the player about the cards certain to be in the opponent's hand or to be a conservative safe discard. First, we will introduce the Plug Rule for safe discards.

A given card A is considered a safe card if all of the following conditions are satisfied:

- The opponent has discarded at least one card of the same rank as card A, or the player knows that at least two cards of the same rank as card A are not in the opponent's hand.
- $\{A, B, C\}$ is a run and at least one of B or C is known not to be in the opponent's hand. E.g. consider 6♣, for which there are three possible containing runs: $\{4♣, 5♣, 6♣\}$, $\{5♣, 6♣, 7♣\}$, and $\{6♣, 7♣, 8♣\}$. The card 6♣ is safe if, in each of these runs, at least one of the other cards is known to not be in the opponent's hand.

Second, we provide the Plug Rule for known cards in the opponent's hand. If the opponent picked up a face-up card B, we mark B as known and assume that B is part of a set (or run) if it is possible and if the player knows that B cannot make a run (or set) in the opponent's hand. The cards that B melds with are assumed to be in the opponent's hand and are marked as known cards.

We implemented this Plug Rule and created a 4×13 array with row and column corresponding to suit and rank, respec-

tively. This *Shankar array* kept track of each card, where each cell was numbered as follows:

- -2 if the card is in the players' hand,
- -1 if the card is a safe discard or is not in the opponent's hand,
- 1 if the card is in the opponent's hand or likely to be in the opponent's meld, and
- 0 if the card is unknown.

For example, if the player's hand consists of $\{A\clubsuit, A\spadesuit, 5\diamondsuit, 6\clubsuit, 6\spadesuit, 7\spadesuit, 8\spadesuit, J\clubsuit, Q\diamondsuit, K\spadesuit\}$ and the face-up card is $K\heartsuit$, the Shankar array for the opponent's hand is as follows:

	A	2	3	4	5	6	7	8	9	T	J	Q	K
C ♣	-2	0	0	0	0	-2	0	0	0	0	-2	0	0
H ♡	0	0	0	0	0	0	0	0	0	0	0	0	-1
S ♠	-2	0	0	0	0	-2	-2	-2	0	0	0	0	-2
D ♢	0	0	0	0	-2	0	0	0	0	0	0	-2	0

After, the opponent has declined the face up card $K\heartsuit$, the table remains unchanged since we knew earlier that $K\heartsuit$ was not in the opponent's hand. However, now the Plug Rule comes to play. Let us observe that $K\heartsuit$ and $K\spadesuit$ are not in the opponent's hand and $Q\diamondsuit$ is in our hand. Using, the Plug Rule, we have that $K\diamondsuit$ is a safe discard, so we can mark it as safe. Similarly, using the same rule, we can conclude that $K\clubsuit$ is also a safe discard. Thus, we get the following table:

	A	2	3	4	5	6	7	8	9	T	J	Q	K
C ♣	-2	0	0	0	0	-2	0	0	0	0	-2	0	-1
H ♡	0	0	0	0	0	0	0	0	0	0	0	0	-1
S ♠	-2	0	0	0	0	-2	-2	-2	0	0	0	0	-2
D ♢	0	0	0	0	-2	0	0	0	0	0	0	-2	-1

Then, the opponent discards an $8\clubsuit$ and the player does not pick it up, so $8\clubsuit$ becomes -1 in the Shankar array, since we know that the opponent does not contain it; afterwards, we pick $7\clubsuit$ up, and the table looks as follows:

	A	2	3	4	5	6	7	8	9	T	J	Q	K
C ♣	-2	0	0	0	0	-2	-2	-1	0	0	-2	0	-1
H ♡	0	0	0	0	0	0	0	0	0	0	0	0	-1
S ♠	-2	0	0	0	0	-2	-2	-2	0	0	0	0	-2
D ♢	0	0	0	0	-2	0	0	0	0	0	0	-2	-1

Then, the player discards $7\clubsuit$, and the opponent picks it up. Since we know that the opponent does not contain $6\clubsuit$ and $8\clubsuit$ and the player knows nothing about $7\heartsuit$ and $7\diamondsuit$, then using the Plug Rule, we assume that the opponent picks $7\clubsuit$ to make a set of 7s, so we get the following table:

	A	2	3	4	5	6	7	8	9	T	J	Q	K
C ♣	-2	0	0	0	0	-2	1	-1	0	0	-2	0	-1
H ♡	0	0	0	0	0	0	1	0	0	0	0	0	-1
S ♠	-2	0	0	0	0	-2	-2	-2	0	0	0	0	-2
D ♢	0	0	0	0	-2	0	1	0	0	0	0	-2	-1

This is an excerpt of the use of the Plug Rule in the Shankar array. These 4×13 Shankar arrays are then passed on to the Convolutional Neural Network.

Convolutional Neural Network We decided to use a CNN in order to detect visual patterns on the Shankar array that might correspond to melds. However, noting that the order of the rows (i.e. suits) can obscure the detection of

Ambi-Superpermutations

Definition 1. *An* ambi-superpermutation on n elements (n-ASP) *is a sequence of* $\{1, \ldots, n\}$ *that contains each permutation of* $\{1, \ldots, n\}$ *as a continuous subsequence ("subpermutations") in some direction.*

For example, 12312 is a 3-ASP, since it contains 123, 132, and 213 as subpermutations. It is useful to draw a diagram that marks each time a character in an ASP is used in a subpermutation. The "star-diagram" for the ASP 12312 follows.

	*	*	*	*
*	*	*	*	
*	*	*		
1	2	3	1	2

Observe that

$$12341231423124312$$

is an 4-ASP of length 17 and has the following star-diagram.

		*		*	*	*		*		*	*	*		*		*
	*	*	*	*	*		*		*	*	*		*		*	
*	*	*	*	*		*		*	*	*		*		*		
1	2	3	4	1	2	3	1	4	2	3	1	2	4	3	1	2

Lemma 2. *If P is a 4-ASP containing two adjacent elements a and b that are both used in 4 subpermutations, then P contains the subpermutation $bcdabcda$, where $\{a, b, c, d\} = \{1, 2, 3, 4\}$.*

Proof. Without loss of generality, let $a = 1$ and assume a precedes b. As a must be the first element in a subpermutation, we can assume $a \neq b = 2$, and that the following two elements are 3 and 4. Since b must be the first element of a subpermutation, the next element after 4 is 1; a must be the second, third, and fourth element of a subpermutation, so the three elements before a are 2, 3, and 4. Thus $bcdabcda = 23412341$. □

Theorem 3. *The smallest 4-ASP has a length of 17.*

Proof. If we consider the star-diagram for the 4-ASP above, we see that each of the 12 permutations of 4 elements only appears once, and that there are 3 element's columns that contain 4 stars; the diagram has a total of 48 stars. It is sufficient to prove that there is no 4-ASP of length 16, for this then proves there is no 4-ASP of length less than 16.
Suppose P is a 4-ASP of length 16, so by the pigeon hole principle, the star-diagram of P contains two adjacent elements with 4 stars each. Then by Lemma 2, P contains two adjacent and repeating subpermutations, so its star-diagram must contain (exactly) $48 + 4 = 52$ stars. Then without loss of generality, $P = 1234123412341234$, contradicting that P is a 4-ASP. □

sets, we were motivated to try a novel approach of creating an augmented representation that makes it possible to detect shapes and patterns that may be relevant for set recognition and related patterns. We compactly ensured that every permutation of the four suits appeared in the order of the rows. The notion of a "superpermutation" was helpful in guiding our decision to modify and extend the Shankar array, but since we only required that every permutation was included in the row ordering in *some* direction, we modified the concept, and developed the idea of an "ambi-superpermutation" (ASP) to fit our needs.

We verified with brute force computation that the minimal size of a 4-ASP is 17, which is about half the size of the minimal size superpermutation on 4 elements which has length 33. This consideration of symmetry reduced the complexity of the array and the computation time.

Experiment Design

It is important to note that since there are many ways to play Gin Rummy, without assuming some sense of how your opponent makes decisions, it is not possible to gain any insight into their hand. Therefore, when comparing the algorithms outlined above, we study a simple player that follows three rules:

- Pick a face-up card only if it makes a meld.
- Choose a discard card randomly from cards that maximally decrease deadwood points.
- Knock as soon as possible, i.e. deadwood points less than or equal 10.

In order to judge the three methods of hand estimation, we define five metrics that will evaluate their effectiveness by comparing the predicted probability distribution and the actual opponent's hand (a probability distribution of 0s and 1s).

The estimators are used independently, but simultaneously, on 50,000 simulated games between two simple players. Clearly, as each game progresses, the estimators will get more information about the opponent's hand, so are expected to produce different metric scores. Therefore, we will track each estimator's metric scores over the course of each game.

Metric Design

Mean Squared Difference The mean squared error is simply the average coordinate value of the squared coordinate-wise difference between the two arrays. This is a considerably basic measure of correlation; the smaller the mean squared error, the better the approximation.

Probabilistic Integrity Loss Since the opponent is always holding 10 cards, the sum of the predicted probabilities should sum to 10. The Probabilistic Integrity Loss measures the percent difference the sum of the predicted probabilities is from 10; the lower the Probabilistic Integrity Loss is, the better the approximation.

Minimum Top Largest The Minimum Top Largest metric counts the minimum number of highest probability entries in the predicted probability opponent hand array so that the 10 cards that are actually in the opponent's hand are included. The minimum value of this metric is 10, and lower values rank the estimation higher.

Top n Cards This metric counts the number of cards that are in the opponents hand and have one of the ten largest predicted probabilities. This metric gives an idea of how well the estimator guesses the opponent's hand in completion. The higher this metric is, the better the estimator is. This is similar to "precision at n" for information retrieval (Craswell 2009).

Drift Area We use the Drift Area metric to encapsulate how accurate an estimator's prediction is of the k most probable cards in the opponent's hand, for all $k \in \{1, \ldots, 10\}$. The algorithm to compute the Drift Area d is as follows:

- Initialize i and d to 0.
- For each $k \in \{1, \ldots, 10\}$,
  - while the number of the top i probability cards is less than k, increment i by 1;
  - add the area $(11 - i)(i - k)$ to d.

This metric penalizes for errors in predicting one of the "best" guesses more than errors in predicting one of the lesser certain probabilities. This is related to "discounted cumulative gain" for information retrieval (Järvelin and Kekäläinen 2009).

Experimental Results

The following plots show the data collected at each turn (after the turn decisions are made) from 50,000 games between two simple players. The values of N that are shown along the bottom of each plot are the number of samples averaged for each data point (the number of times a simulated game reached that turn).

The shaded region shows the standard error of each average and is generally only discernible towards the maximum turn number. In each plot, red, blue, and green shows the performance of the Simple Neural Network, Bayesian, and Shankar CNN estimators, respectively.

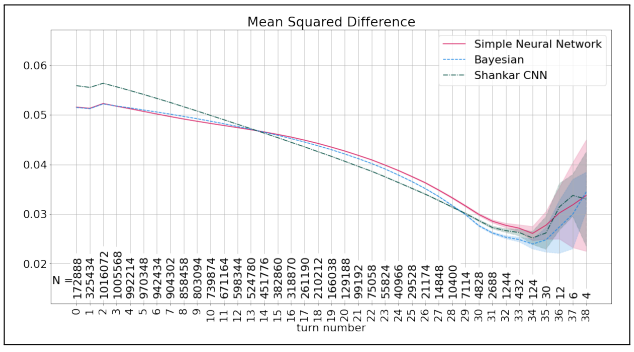

Figure 2: Mean Squared Difference vs. Turn Number

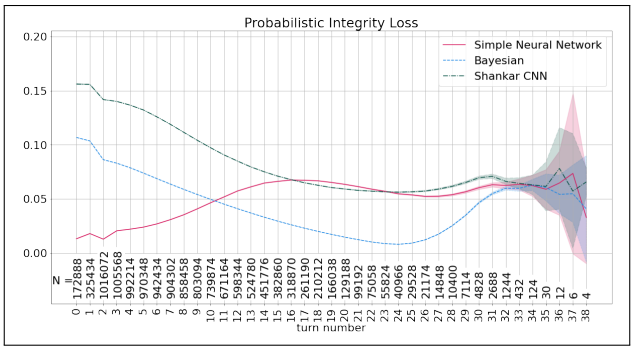

Figure 3: Probabilistic Integrity Loss vs. Turn Number

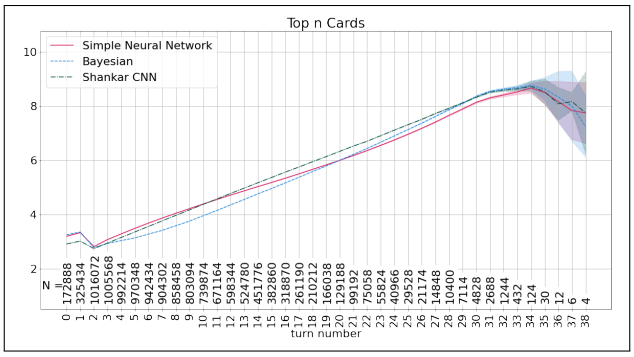

Figure 5: Top n Cards vs. Turn Number

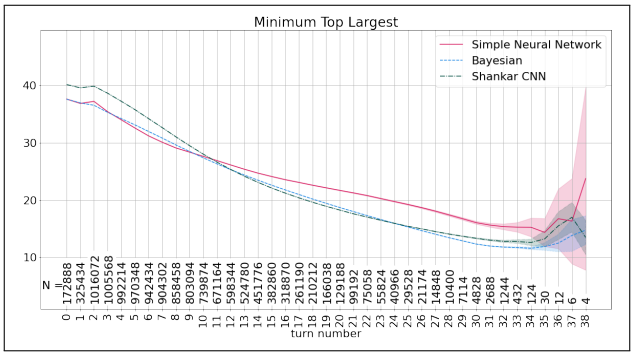

Figure 4: Minimum Top Largest vs. Turn Number

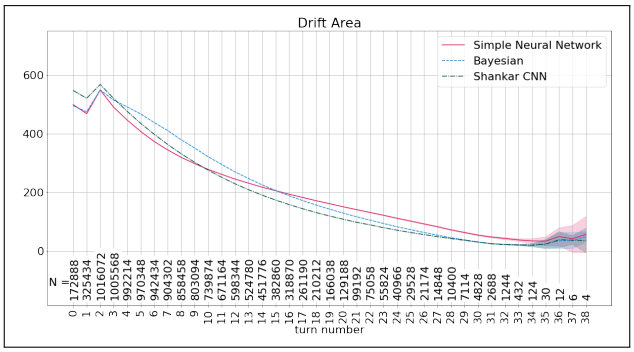

Figure 6: Drift Area vs. Turn Number

By the Mean Squared Difference metric, the Simple Neural Network and Bayesian estimators are nearly evenly matched in the early game. Shankar CNN pulls ahead in mid game and by late game, Bayesian narrowly beats the other two (Fig. 2).

By the Probabilistic Integrity Loss metric, the Simple Neural Network Hand Estimator is closer to 0 than other two hand estimators until turn 10, after turn 10 Bayesian Hand Estimator is closer to 0 than other two hand estimators. After turn 16, Simple NN Hand Estimator and Shankar CNN Hand Estimator are matched until the end of the game (Fig. 3).

By Minimum Top Largest metric the Simple Neural Network Hand Estimator and Bayesian Hand Estimator are better than Shankar CNN. After turn 12, Shankar CNN and Cayesian are matched up with a little advantage for Shankar up to turn 24 and with an advantage for Bayesian after turn 24. The Simple Neural Network performs significantly worse than both of the other estimators after turn 10 (Fig. 4).

By the Top n Cards metric, the Simple Neural Network estimator is preferable in the early game; by the mid game through the end, the Shankar CNN remains to be the better estimator (Fig. 5).

By the Drift Area Metric, the Simple Neural Network is the better estimator in the early game, and then the Shankar CNN pulls ahead until the end of the game (Fig. 6).

We observe that each estimator has periods of the game where it excels in one or more metrics for success. By providing these various estimators, a player could switch between using the one expected to be most advantageous for each situation in the game. In the early game (up to the first 3 turns) we recommend a simple neural network or the Bayesian hand estimator. Then, in the second stage of the game, from turn 4 to turn 9, one should switch to the simple neural network. In the middle game, between turn 10 and turn 28, we recommend changing to the Shankar CNN estimator. Lastly, from turn 29 until the endgame, either stick to Shankar CNN or change to the Bayesian hand estimator. Since each of these estimators are time efficient, altering between them will not significantly increase the computation time. Thus, with no one estimator consistently dominant, an ensemble of estimators is recommended at this time.

Hand Estimation Application

There are multiple ways that hand estimation can be adopted to the Gin Rummy player's strategy; it can be used not only to decide which card to discard safely, but it may also be used to decide whether to pick up the face-up card or not. Moreover, the hand estimation is able to help the player judge the possibility of the opponent either going for a gin or for a low deadwood score which further influences the player's knocking strategy. We selected a simple strategy for deciding a safe discard to show the potential benefit of using hand estimation in Gin Rummy.

We used a Simple Player against a modified a Simple Player, which played with a modified discard rule:

Among the top three safest discards, discard the card that decreases the deadwood score the most.

In deciding the safest discard card, this modified player used the Shankar CNN Hand Estimator, and after playing against the Simple Player for $100,000$ games, it won approximately 55% of all matches. This small implementation of hand estimation has shown already an advantage over the Simple Player.

Future Work

Here, we have implemented approaches to opponent hand estimation, and have applied it for one simple enhancement to decision-making. However, hand estimation can be used in more intricate and diverse strategies at each player's decision, i.e. whether to pick up a face up card, discard a card, knock, or go gin. Moreover, we would like to improve the hand estimator by improving adaptability: a player that dynamically learns which of the hand estimators performs better against a given play style. A hand estimator that could better estimate an opponent's strategy could better predict opponent card probabilities. However, as with the case of DeepStack, this approach would require more computation. Therefore, more work has to be done on how to reduce time complexity of the hand estimation.

Furthermore, we limited our work to the study of a player that makes relatively simple decisions. The same techniques could be applied to train on, compete, and aid more advanced players. We predict that with a stronger player, the CNN hand estimation especially could be more effective, as it trains its estimation model from a better play.

The application of the estimation was not fully developed here, so another avenue for continued work is to compare the effectiveness of the player that uses the estimation. We would like to see where our estimator falls between the extremes of a trivial "null" estimator with equiprobable estimates and an "omniscient" estimator that can cheat and report complete information of the opponent's hand.

Conclusions

In this paper, we have introduced and evaluated some methods for opponent hand estimation. With the use of Shankar's Plug Rule, we have built a Convolutional Neural Network that estimates the opponent's hand through pattern recognition. To improve the readability, we developed the theory of an ambi-superpermutation in order to efficiently pre-process the input and represent the hand as a CNN image input with the suit-rows of data in a Shankar Array. With a simple application of such hand estimation to a simple player's discard decisions, we observed a statistical win-rate advantage of 5% against the player without such hand estimation.

By comparing different approaches, we concluded different estimators dominate at different turn ranges of the game. Thus, we would recommend an ensemble of our estimators: use a simple neural network or the Bayesian hand estimator during the first 3 turns, use the simple neural network between turns 4 and 9, between turns 10 and 28 use the Shankar CNN estimator, and use the Shankar CNN or the Bayesian hand estimator after turn 29.

Acknowledgments

This work was supported, in part, by the Cross-Disciplinary Science Institute at Gettysburg College (X-SIG).

References

BoardGameGeek.com. 2020. Most popular standard deck card games. URL https://tinyurl.com/bggstdcardgames. Accessed: 2020-12-12.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359(6374): 418–424. ISSN 0036-8075. doi: 10.1126/science.aao1733. URL https://science.sciencemag.org/content/359/6374/418.

Craswell, N. 2009. Precision at n. URL https://doi.org/10.1007/978-0-387-39940-9_484.

Järvelin, K.; and Kekäläinen, J. 2009. Discounted Cumulated Gain. URL https://doi.org/10.1007/978-0-387-39940-9_478.

McLeod, J. 2020. Gin Rummy - Card Game Rules. URL https://www.pagat.com/rummy/ginrummy.html. Accessed: 2020-12-12.

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337): 508—-513. ISSN 1095-9203. doi:10.1126/science.aam6960. URL http://dx.doi.org/10.1126/science.aam6960.

Parlett, D. 2008. *The Penguin Book of Card Games*. Penguin. ISBN 9780141037875. Accessed: 2020-12-12.

Parlett, D. 2020. GIN RUMMY "The game of the stars". URL https://www.parlettgames.uk/histocs/ginrummy.html.

Ranker. 2020. The Most Popular Fun Card Games. URL https://www.ranker.com/crowdranked-list/most-fun-card-games. Accessed: 2020-12-12.

Rollason, J. 2007. Predicting Game States in Imperfect Information Games. URL https://www.aifactory.co.uk/newsletter/2007_02_imperfect_info.htm.

Shankar, P. 1994. *How to Win at Gin Rummy: Playing for Fun and Profit*. Brattleboro, Vermont, USA: Echo Point Books & Media. ISBN 9781626541979.