Random Forests for Opponent Hand Estimation in Gin Rummy

Anthony Hein[1], **May Jiang**[1], **Vydhourie Thiyageswaran**[1], **Michael Guerzhoy**[1,2,3]
[1] Princeton University
[2] University of Toronto
[3]Li Ka Shing Knowledge Institute
anhein@princeton.edu, mayjiang@princeton.edu, vrt2@princeton.edu, guerzhoy@cs.toronto.edu

Abstract

We demonstrate an AI agent for the card game of Gin Rummy. The agent uses simple heuristics in conjunction with a model that predicts the probability of each card's being in the opponent's hand. To estimate the probabilities for cards' being in the opponent's hand, we generate a dataset of Gin Rummy games using self-play, and train a random forest on the game information states. We explore the random forest classifier we trained and study the correspondence between its outputs and intuitively correct outputs. Our agent wins 61% of games against a baseline heuristic agent that does not use opponent hand estimation.

Introduction

Gin Rummy is an imperfect-information card game. The rules of the game, as well as Todd Neller's baseline player, are available at the EAAI 2021 Gin Rummy Challenge website[1].

Optimal play likely requires reasoning about the opponent's strategy as well as reasoning about, for example, which cards are in play. Instead of using only explicit reasoning, the agent we describe uses a statistical model that predicts the opponent's hand together with very simple reasoning in order to choose plays.

We fit a statistical model to game information states generated using self-play of a simple heuristic model. Specifically, we learn to predict, for each card, the probability that it is in the opponent's hand. The agent uses this information as part of its strategy.

Statistical models are widely used in game-playing agents instead of, or in conjunction with, explicit reasoning. For example, AlphaGo (Silver et al. 2016) learns a value network, which assigns scores to game states that correspond to how good the states are for the agent. In (Veness et al. 2009), the parameters of a heuristic evaluation function are learned from game data. Learned models have been used to evaluate properties of states in card games (Sang and Yoon 2019).

Kotnik and Kalita (2003) train an agent to play a version of Gin Rummy using temporal difference learning (Sutton and Barto 2018). They learn a feedforward neural network (ANN) as a value function. The ANN takes as an input a 52-dimensional encoding of the information state, with each dimension corresponding to one of the cards in the deck, with positive input values corresponding to possession of the card, and negative values corresponding to the inaccessibility of the card to the agent. Our approach also utilizes a learned non-parametric function of the information state. However, our function is used for estimating the opponent's hand rather than the value of the information state.

We will first describe a baseline Gin Rummy agent due to Todd Neller. We will then describe and evaluate our model for estimating the opponent's hand, show how the model can be used to improve play, and evaluate the agent that uses the model when playing. Our agent builds on the baseline agent by incorporating the model's predictions. Finally, we analyze the model to show that it works as one would expect some of the time, but its outputs sometimes diverge from what we would expect some of the time.

A Simple Agent for Gin Rummy

In this Section, we describe a heuristic-based Gin Rummy agent due to Todd Neller; our agent both builds on and is evaluated against this baseline agent which will be explained in the paragraphs to follow.

When prompted to draw a card, the baseline heuristic agent will draw the face-up card if and only if it could be melded with other cards in the agent's hand. If the face-up card appears in no prospective melds, the agent will instead draw the face-down card. In doing so, the agent focuses itself on melding cards so that it can get closer to knocking, and avoids having a situation where the opponent knows which cards the agent is holding.

To discard a card, the agent will consider all possible hands created from the removal of one card and observe the deadwood achieved with each possible hand (assuming the best meld sets are played). The agent will then randomly discard a card from the collection of candidate cards whose removal leaves a hand with minimal deadwood. There are two exceptions to this heuristic: the agent will not discard a card they have just drawn from the face-up pile; and the agent will not discard a card if it means that they will have drawn and discarded the same card twice within the same game. This strategy attempts to minimize the deadwood if the opponent knocks at the next turn.

[1]http://cs.gettysburg.edu/~tneller/games/ginrummy/eaai/

The baseline heuristic agent will knock as soon as possible. When knocking, the agent will randomly select a best meld set to report as its final melds. Generally, as the game goes on the opponent's deadwood will decrease, making it a fair strategy to knock as soon as possible to avoid the possibility of being undercut.

Random Forests for Opponent Hand Estimation

We perform opponent hand estimation by assigning a score to each card, with a higher score corresponding to a higher probability that the card is currently in the opponent's hand. More specifically, given as input the information about the current player's hand, the draws, and the discards made thus far, the goal of the model is to output, for each of the 52 cards in the deck, the probability that the card is in the opponent's hand. We use multi-output random forests (Dumont et al. 2009) trained on games generated using self-play to achieve this.

Multi-Output Random Forests

A random forest is an ensemble of decision trees, with each decision tree trained on a subset of the training data. For classification problems, at test time, all decision trees are run on the input, and the output is the fraction of decision trees that output 1, and can be thought of as the predicted probability of 1.

Decision trees can be extended to multi-output settings (Dumont et al. 2009). In a multi-output setting, the output we are attempting to predict is a k-dimensional vector of 1's and 0's. When training a decision tree in this setting, we choose splits that minimize the average Gini impurity for the k output dimensions. For $k = 1$, this reduces to choosing splits that minimize the Gini impurity for the output.

In our case, we have $k = 52$ outputs. The i-th output's being 1 corresponds to the i-th card's being in the opponent's hand. The i-th output is 0 otherwise.

We use scikit-learn (Pedregosa et al. 2011) to train our random forest. We set the minimum number of samples for a leaf to 2 to reduce overfitting, and train 100 trees. To speed up training of trees, the algorithm considers at most $\sqrt{152} \approx 12$ features for each split.

Training the Opponent Hand Estimator

We generated a dataset of information states and corresponding opponent hand encodings through self-play by the baseline heuristic agent. After running $20,000$ such games with random deck initializations, and recording after each play – defined as each end of a player's turn to draw and discard, excluding the initial face-up card plays – information about the current player's hand, the drawn and discarded cards, and the opponent's hand, the dataset contained $2,571,176$ plays.

For each play, the information state encoding – the input to the model – was constructed as a 3×52 array of one-hot encodings of discarded cards, cards picked up by the opponent, and cards in the current player's hand. Note that discarded cards include cards discarded by the opponent as well as cards discarded by the current player that the opponent did not opt to pick up. See Fig. 1. The corresponding output was the one-hot encoded hand of the opponent, as a 52-dimensional array.

	A♡	A♢	...	10♡	...	...	K♣
Discarded	0	1	...	0			1
Picked up	1	0	...	0	...	...	0
In hand	0	0	...	1	...	...	0

Figure 1: Encoding of an information state. The Ace of Diamonds and the King of Clubs were discarded. The opponent picked up the Ace of Hearts. The 10 of Hearts is in the agent's hand. The input vector for the random forest is of length $3 \times 52 = 156$

We trained the classification model on a random subset of $100,000$ of the plays from $10,000$ of the games, and used a separate, held-out, random $100,000$ of the plays from a separate and independent set of $10,000$ games to test the model.

Evaluating Opponent Hand Estimation Models

To evaluate the model, we used the standard classification metrics of accuracy, precision, recall, and the F1 score. For each of the 52 cards in the deck, the binary classification model generates for each of the two classes the predicted probabilities that the card is in each class, and the predicted probability that the card is in the positive class can be interpreted as the probability that the card is in the opponent's hand. The accuracy then measures the proportion of correct predictions out of all predictions, where a prediction for a given card c is correct if the model outputs $P(Y_c = 1) > t$ and the card is in the opponent's hand or $P(Y_c = 1) \leq t$ and the card is not in the opponent's hand. The parameter t is a probability threshold that represents a tradeoff between precision and recall. We find that a threshold close to $t = 0.25$ maximizes the F1 score (see the next section).

Baseline Classifiers

We compared our classification model to two baselines: first (B1), we simply select ten cards from the deck at random to be in the opponent's hand and classify the rest to be not in the opponent's hand. Second (B2), we predict ten cards from the deck to be in the opponent's hand, but incorporate the information about discarded and picked up cards known to the current player. In particular, for cards that are known to be in the opponent's hand – face-up cards picked up by the opponent that have not been discarded – we predict that those cards are in the opponent's hand, and for the remaining up to 10 cards of the opponent's hand, we select at random from the remaining cards in the deck that have not been discarded. The performance of our random forest classifier on $10,000$ held-out game state-opponent hand pairs from the randomly generated games in the test set – independent of the games used in training – is compared to that of these two baselines in Table 1. The random forest model outperforms both baselines in every metric, and with an F1 score of 0.67

Model	Accuracy	Precision	Recall
B1 (no information)	0.6906	0.1956	0.1956
B2 (with information)	0.7407	0.3257	0.3257
Logistic Regression (t=0.25)	0.7154	0.4057	**0.8102**
Logistic Regression (t=0.3)	0.7648	0.4393	0.6507
KNN (k=5)	0.8023	0.4998	0.6666
Random Forest (t=0.25)	0.8330	0.5821	0.7889
Random Forest (t=0.3)	**0.8796**	**0.6852**	0.6266

Table 1: Opponent Hand Estimation Performance

at a threshold of 0.25, outperforms both a logistic regression model and a K-Nearest Neighbors classifier trained and tested on the same data, with F1 scores of 0.54 and 0.57, respectively. The comparably poor performance of the logistic regression classifier could be attributed to the weakness of a linear model for learning this complex task. We note, moreover, that predicting all zeros – predicting that every card is not in the opponent's hand – would result in an accuracy of $1 - \frac{10}{52} = 0.8$, and that the random forest model outperforms this as well.

Opponent Hand Estimation to Improve Play

We use Opponent Hand Estimation (OHE) in deciding which cards to discard. Our approach uses OHE by estimating which cards are more likely to be drawn from the draw pile. That is, we compute the quantity $p_c^{(d)}$ for a card c that is potentially in the draw pile, with $p_c^{(d)} = 1 - p_c^4$, where p_c is the probability of card c being in the opponent's hand according to our random forest model. We compute a score w for each card, which represents the usefulness of the card in forming melds.

For a card c_1, together with one other card c_2 from the agent's hand, we consider all the ways in which c_1 and c_2 could form a meld with an unseen card, for every c_2. When computing the weighted number of ways that c_1 could be used to form a meld, we weight each potential meld by $p_{c_3}^{(d)}$, with c_3 being a card that could be in the discard pile. This weighted number of ways that c_1 could be used in forming melds represents the usefulness of c_1.

The quantity $p^{(d)}$ is computed using $1 - p^4$ to pull the values of $p^{(d)}$ toward 1. We found that the model tends to have p be close to 0.2. We are using $p^{(d)}$ as an estimate of the probability that the card is in the draw pile, and found that performance increases when larger $p^{(d)}$s are close to 1.

For each card in the agent's hand, we compute the deadwood points d of the hand if this card were discarded, similarly to the baseline agent. In choosing the card to be discarded from a hand, we use a linear combination of deadwood points d associated with discarding the card and the score w the card. That is, for each card c and its associated deadwood points d and score w, we compute the value v:

$$v = w \times \alpha + d \times (1 - \alpha)$$

We determine the value of α through a parameter search. At $\alpha = 0.1$, the win rate against the baseline agent is 0.6. See Fig. 2.

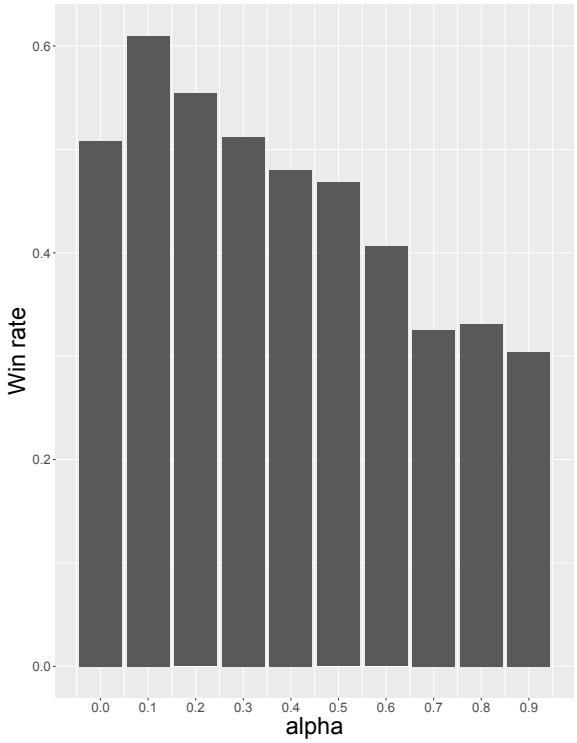

Figure 2: The win rate for different weightings of the melding score w and the deadwood points d, out of 1000 games.

The agent discards the card with the lowest v. We want to trade off the amount of deadwood points that accrue due to discarding the card (hence we minimize d) and the number of melds that we could not form due to discarding the card (hence we want to minimize w).

In evaluating the OHE model, we ran a tournament of 1000 games between the baseline agent and our OHE agent. The OHE agent won 610 games (61.0%) against the baseline player.

What Does the Random Forest Classifier Do?

In this Section, we explore the random forest predictor to see whether its functionality corresponds to what one would expect. This Section is structured as a series of experiments: we posit hypotheses about the behavior of the predictor, and then confirm or disconfirm them.

We usually look at the difference between the outputs of the random forest due to a change in the agent's hand or the discard pile. The differences are usually small, but statistically significant, since we use a large number of random hands. Note that we transform the outputs of the random forest when using them, amplifying the effect of the small differences.

The outputs of the random forest are estimates for probabilities, and we refer to them as such. However, as we shall see, the probabilities are not calibrated, and should be thought of instead as quantities that are positively correlated with probabilities.

Ace	0.02 (p <0.01)
2	0.03 (p <0.01)
3	0.02 (p <0.01)
4	0.02 (p <0.01)
5	0.02 (p <0.01)
6	0.01 (p <0.01)
7	0.01 (p <0.01)
8	-0.01 (p <0.01)
9	-0.06 (p<0.01)
10	-0.02 (p<0.01)
J	-0.02 (p<0.01)
Q	0.002 (p = 0.2)
K	-0.01 (p <0.01)

Figure 3: Changes in the average probability for the opponent's holding a card of rank R when replacing a Queen with a 9 in the list of cards discarded by the opponent

Discarding a higher-rank card means having lower-rank cards. We expect that if the opponent discards a card of rank R, they do not have many cards of rank $> R$, because the algorithm will discard the highest-ranking card that's not in a meld.

We confirm that the random forest agrees with this intuition with an experiment. For 250 random hands, we compare the average probability of holding a card of each rank when adding a 9 or a Queen to a list of four cards discarded by the opponent. The results are in Fig. 3. If we see the opponent discarding a 9 rather than a Queen, the estimated probability of the opponent's holding lower-ranked cards increases, and the probability of them holding higher-ranked cards decreases. For each rank R, we average the probabilities the random forest outputs across all four suits. We indicate the p-value in parentheses. Our sample of random hands is large enough for all the changes to be significant.

Opponent's discarding a card of a certain rank means opponent is less likely to hold cards of the same rank. If the opponent discards a card of rank R, we expect that they do not have multiple other cards of rank R, because otherwise they could potentially be melded. We confirm that for 250 random hands, replacing a card of rank 10 discarded by the opponent with a card of rank 5, the average probability across all suits that the opponent holds cards of rank 5 in their hand decreases by 0.009 ($p < 0.001$).

Opponent's picking up a face-up card of a certain rank makes them more likely to hold cards of the same rank. Conversely, we expect that if the opponent chooses to pick up a face-up card of rank R from the discard pile, the opponent's hand likely contains other cards with which the picked-up card would form a meld. Because one way to form a meld is to combine at least three cards of equal rank, one possibility is that the opponent's hand contains other cards of the same rank as that of the face-up card. We confirm that for 250 random hands, replacing a card of rank 10 picked up by the opponent with a card of rank 5, the average probability across all suits that the opponent holds cards of rank 5 in their hand increases by 0.03 ($p < 0.001$). Even ignoring the picked-up card that is now known to be in the opponent's hand, the average probability across all of the other suits that the opponent holds cards of rank 5 increases by 0.026 ($p < 0.001$).

Opponent picking up a card of a certain rank makes them more likely to hold cards of adjacent ranks. Since a meld can also be formed by combining three or more cards of the same suit with consecutive ranks, we expect that if the opponent picks up a card of rank R from the discard pile, they are more likely to hold cards of ranks adjacent to R with the same suit that could be used to complete a consecutive-rank meld. Cards of the same suit with rank $R-2$, $R-1$, $R+1$, or $R+2$ are most likely to serve this purpose.

To verify that the model captures this intuition, we again replace a card of rank 10 picked up by the opponent with a card of rank 5 for 250 random hands. As expected, we find that when the opponent picks up the card of rank $R = 5$ rather than rank 10, the probability that the opponent also holds the card of rank $R+1 = 6$ from the same suit increases by 0.02 ($p < 0.001$) and the probability that the opponent also holds the card of rank $R + 2 = 7$ from the same suit increases by 0.006 ($p = 0.0017$). The probability that the opponent also holds the card of rank $R + 3 = 8$ from the same suit instead decreases by 0.07 ($p < 0.001$), and the probability that the opponent also holds the card of rank $R+4 = 9$ from the same suit decreases by 0.1 ($p < 0.001$).

Noting that 6 and 7 are four and three ranks below rank 10, respectively, while 8 and 9 are two ranks and one rank below rank 10, respectively, these results agree with our intuition: the cards of ranks between 5 and 10 that are adjacent to and more likely to be melded with the card of rank 5, are predicted to be in the opponent's hand with higher probability when the opponent picks up the card of rank 5, whereas the cards that are adjacent to and more likely to be melded with the card of rank 10 are predicted to be in the opponent's hand with higher probability when the opponent picks up the card of rank 10 instead.

The agent's holding a card of a given rank makes the opponent less likely to hold a card of the same rank. If I am holding an Ace, there are fewer Aces for my opponent. We confirm this observation with an experiment: in 1000 random hands, we replace one of the agent's cards with a card of a different rank. We observe a decrease in the estimated probability of the opponents' holding a card of rank R if the agent holds a card of rank R by 0.015 ($p < 0.001$).

Probabilities should approach 0 or 1 during the endgame. If we have seen all the cards, the probabilities for unseen cards should approach 0 and 1. In fact, that does not happen. The random forest estimates almost all probabilities to be close to 0.2 (see Fig 4). This may be an artifact of the fact that we train the random forest on all game states, most of which are not at the endgame. Note that, for 52 cards, of which 11 are known to definitely not be in the opponent's hand (the agent has 10 in its hand, and sees the face-up card), and without any further information (i.e., at the beginning of the game), the probability of an unseen card being in the opponent's hand is $10/41 = 0.24$.

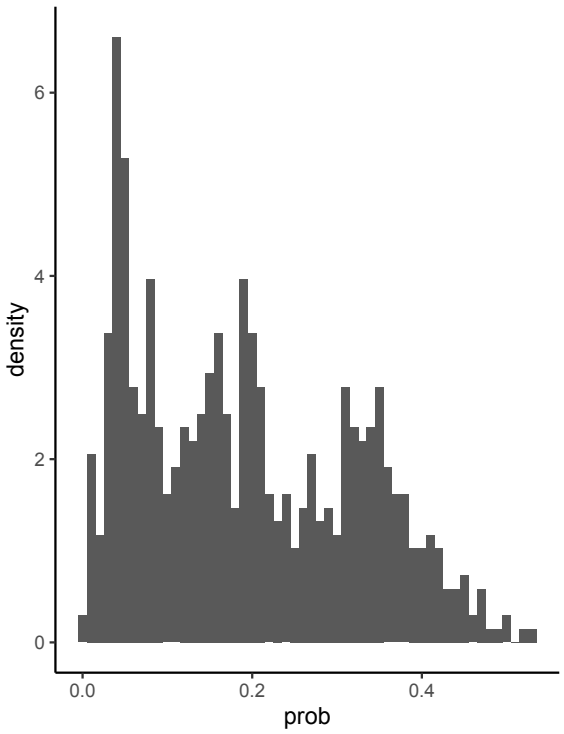

Figure 4: A density plot of the probabilities that the card is in the opponent's hand, for each card, at the endgame. Data collected from 100 random game states.

Probabilities do not go to 0 or 1 in the middle of the game. In a random hand, having all of the 2's be in the discard pile only decreases the estimated probability that the opponent has a 2 by 0.025. This again may be explained by not having enough endgames in the dataset.

The probabilities are roughly symmetric with respect to suit. It should be the case that, if for example Diamonds are switched with Spades throughout, the probabilities for all cards remain the same. That is roughly the case. In 500 random game states, we switched two random suits in the game state. We then compared the probability distribution over the opponent's cards evaluated using the game state with the suits switched to the distribution over the opponent's cards when the suits are not switched. The median correlation between the probabilities obtained in the two ways is 0.84, indicating a substantial agreement. The correlation should reach 1.0 with a very large training set, since the agent used to generate the training set treats all suits the same.

In future work, we will compute the probabilities by computing them using all $4! = 24$ permutations of the suits, and averaging the probabilities across the suits.

Overall assessment. The random forest model we use is a substitute for analytically computing the probability distribution over the opponent's cards by enumerating all possible games. The random forest model captures some, but not all intuitions about the probability distribution over the opponent's cards. It does so without needing to enumerate all possible game trees.

The "probabilities" that the model outputs are informative, but are poorly calibrated. That is part of the reason that we transform them when using them to decide on plays.

In addition, we observed that our model seems to be overfitting. For example, we expect that as the training set size goes to infinity, the model will become exactly symmetric with respect to suit.

Moreover, the model was trained against a particular agent. That means that the outputs of the model will not necessarily work when playing against a substantially different agent.

Conclusions and Future Work

We presented an AI agent that plays Gin Rummy. The agent uses a random forest model to estimate the opponent's hand. Random forests are easy to train and run. We find that they can, to some extent, substitute for explicit reasoning.

We show that the random forest model we trained behaves how an explicit algorithm with reasoning incorporated into it would behave in some situations, but behaves sub-optimally in other situations. This is explained in part by overfitting.

We see several ways that future work could enhance our analysis. First, our opponent hand estimation strategy does not weight recent events more heavily than events at the beginning of a round, even though events later in a round may be more informative as both players improve their hands toward forming melds. One possible strategy to address this is to weight more recent events more heavily when constructing the information state vector.

Further, we compared our random-forest-based opponent hand estimation strategy to simple baselines. Future work includes comparing the random forest model to better heuristic methods for estimating which cards are in the opponent's hand, and possibly combining the heuristic and learning-based approaches.

Finally, our agent is only concerned with our own score when deciding what card to discard. Another direction involves using opponent hand estimation to play in such a way as to prevent the opponent from forming melds.

References

Dumont, M.; Marée, R.; Wehenkel, L.; and Geurts, P. 2009. Fast multi-class image annotation with random subwindows and multiple output randomized trees. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 2, 196–203.

Kotnik, C.; and Kalita, J. K. 2003. The significance of temporal-difference learning in self-play training td-rummy versus evo-rummy. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 369–375.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Sang, B.; and Yoon, S. 2019. A Neural Network Approach for Birds of a Feather Solvability Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 9706–9712.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587): 484–489.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An Introduction*. MIT press.

Veness, J.; Silver, D.; Blair, A.; and Uther, W. 2009. Bootstrapping from game tree search. In *Advances in Neural Information Processing Systems*, 1937–1945.