1 Playing game

1.1 Choosing the right time to knock?

Supposed we are in a gin rummy game and we are able to knock this round. The simplest way is to knock as soon as possible so that don't allow the opponent to optimize their deadwood points. However, knocking as early as possible is not the best strategy. When we reach lower that maximum deadwood point to knock, we are assuming that the opponent is having more deadwood point than us, but what if they are waiting for us to knock for layoff or just try to undercut us?

Therefore, choosing the right time to knock is also one problem that we have to take into account. When we reach the deadwood point that we are able to knock, our decision will be affected by how far the turn have gone, how many meld we have, how many meld the opponent have, and how our melds value be stronger than that of the opponent's melds. For the problem to be simple, we collect each turn data which has number of turn so far, number of melds, and deadwood point, and the label will be whether knocking won or be undercut. The number predicted will be the probability of deciding whether to knock when we reach appropriate deadwood point. We also initialize the threshold that can be learn through out the game. When the predicted value is greater the the threshold, we decide to knock, and vice versa.

	Loss	Acc	Win rate	Increased win rate
Turn=0.5m, Threshold=0.7	0.16	0.83	58%	8%
Turn=1.0m, Threshold=0.8	0.14	0.84	60%	10%
Turn=1.0m, Threshold=0.9	0.14	0.84	62%	12%

Table 1: Training over the simple player

2 Hand Estimation

Hand estimation is a crucial part in estimating player's strategy. When successfully estimating a small portion of opponent's hand, we can choose to discard wisely and safety based on the fact that the discarded card can form a meld with cards that estimated to be in the opponent's hand. With that in mind, we need enough information from each different turn in a match to do our job. Because when we play a turn, we see that opponent does not pick in the discard pile, then we know that they probably do not have any cards that are in meld with out discarded card and vice versa. Additionally, in the discarding phase, we see that the opponent discard a particular card, it can be in the situation

that they do not have any cards that are in meld with that card. Therefore, our information will contain a representation of cards that the opponent does pick/ not pick, and cards that are discarded by them.

Metric: According to the normal accuracy metric, we decide to evaluate the accuracy of estimating a hand in Gin Rummy in 2 ways: categorical accuracy and binary accuracy. For categorical accuracy, the accuracy is measured by summing all accuracy of each card (1 - distance between label and predicted value), and then divide by the number of cards, in this case, 52. The binary accuracy, on the other hand decide each predicted value to be zero or one, and divide the number of true positive and negative with the number of cards.

$$Categorical\ Accuracy = \frac{\sum_{i=0}^{n} 1 - \|c_{actual} - c_{predict}\|}{n}$$

$$Binary\ Accuracy = \frac{\sum_{i=0}^{n} (c_{predict} == c_{actual})}{n},\ threshold = 0.5$$

2.1 LSTM Network

Turn State Representation: There will be 4 inputs. Each is a vector of length 52 (one-hot encoded): vector of cards the opponent discarded this turn, vector of cards that the opponent does not care about (not picking up from discard pile) this turn, vector of cards that the opponent does care and pick up from the discard pile, and cards that are on this player's hand and in the discard pile.

Modeling: The four input collected in the games are feed into the model. First, each input vector go to a lstm cell that remember the predicted opponent hand in relation with the each input. The output of all lstm cells are then concatenated into a single vector. The vector is then fed into the feed forward dense network with dynamic layers.

Loss Function and Gradient Update: Augmented Categorical Cross Entropy:

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c}$$

The problem of this loss function is that it consider all errors at every classes the same. However, in this particular Gin Rummy case, we only want to estimate each card if we have some clue of them in the input. Say it another way, we can only be able to estimate which card the opponent have in hand if we know the existence of some of the cards.

Therefore, to solve this problem, we need to make cards that are relevant to the input cards, and we also want to make unrelated card more trivial. Therefore the formula of the loss function should evaluate to:

$$-W_o * \sum_{c=1}^{M} y_{o,c} \log p_{o,c}$$

	Loss	Acc	Val_loss	Val_acc
DNN	32.39	0.32	35.93	0.29
LSTM	23.25	0.94	23.18	0.92

Table 2: Loss and Accuracy between DNN and LSTM

2.2 Bayes Theorem with Neural Network

Bayes Theorem formula in estimating opponent's hand: When we want to estimate whether a card exist in opponent's hand or not, we base on the sequence of action that the opponent perform previously. In particular, the probability of the opponent have a given card in hand is the chance they have that card given that they have drawn and discarded some particular card in the previous turn. The formula is below:

$$P_t(c^i) = P_{t-1}(c^i|d_{t-1}) * P_{t-1}(c^i|e_{t-1})$$

To calculate these conditional probability, we use Bayes Theorem:

$$P_{t-1}(c^i|d_{t-1}) = \frac{P_{t-1}(d_{t-1}|c^i)P_{t-1}(c^i)}{P_{t-1}(d_t - 1)}$$

and

$$P_{t-1}(c^i|e_{t-1}) = \frac{P_{t-1}(e_{t-1}|c^i)P_{t-1}(c^i)}{P_{t-1}(e_t - 1)}$$

The chance of opponent having a particular card given that they have drawn or discarded some cards is the the multiplication of the chance that the opponent would pick the drawn card or discard the discarded card, with the probability of them having that particular card previously, and then divide by the probability of drawing (1/2), or discarding (1/11). The probability of a card is initialized from the start of the round, and the probability of drawing or discarding given that they have that card that we are estimating, is the meld building pattern of the opponent. To tackle this, we will try to learn the meld building pattern of the opponent by collecting input hand data and drawing and discarding labels. We found that, for simple player, they only draw cards which can form melds, and only discard cards not in meld and minimize deadwood. Additionally, at time t, the chance of a card in opponent hand depends on the sequence of actions in the previous turn, which also depends on the information of every sequence of actions in each turn before. At the start of the game, the estimation matrix is initialized with relative to the unknown cards and is re-calculated in each turn.

2.3 Hand abstraction using VAE

One another approach of defining strategy for a gin rummy player is to go through every situation in the game and continuously traverse the game tree recursively toward the end. For the core of an imperfect information game, we

	Cate_acc	Bin_acc
DNN Model	69.38%	60.85%
LSTM Model	81.07%	80.24%
Bayes System	82.30%	61.54%

Table 3: Metrical Comparison among systems

do not know the reward until we reach the end of the game tree. Only when we do so, we are able to update information set and history with some values saying whether some particular playing paths are winnable or not.

However, the number of information set combinations is very large in Gin Rummy. Throughout the game, for one turn, we can have $52C10$ cards combination posibility in hand, $52C32$ cards in draw pile, and $52C10$ cards combinations in the opponent's hand. We can put it another way: in each turn, if we put player's hand, opponent's hand, draw pile and discard pile each a vector of 52 cards, we would have $4^{52} = 2 * 10^{31}$ possible situation (for each card, one can have 4 different possibilities to be in one of the four card vectors)

Therefore, we need to minimize the number of situation is to abstract the situation and put them into bucket that map to some situation values. We can easily map our calculated value back to the abstracted situation so that when we are in a particular public state, we know which information set we are in and can have proper policy. One way of achieving this is to use an auto-encoder.

Auto-encoder: An auto-encoder consists of 2 parts: encoder and decoder. An encoder consist of neural network layers that reduce dimensionality when the network goes forward, and the decoder takes its input from the reduced encoder's layer and map the features back to an output. Traditionally, autoencoders were used for dimensionality reduction or feature learning. In computer vision, autoencoders are often used in the process of pixelizing or denoising images.

In Gin Rummy in particular, we will apply autoencoder concept by first encoding a public game state with linear neural networks, then we start to decode them to our state label which have been defined to be similar to the input public game state. After that, when we actually traverse the game tree, we start encoding the public game state to a tensor, and have a comparison that map that tensor to a particular bucket information set, to update and evaluate a particular policy.