2020 IEEE 9th Data Driven Control and Learning Systems Conference
November 20-22, 2020, Liuzhou, China

Opponent Modeling in Poker Games

Xi Yan[1], Li Xia*[2], Jun Yang[3], Qianchuan Zhao[3]

1. School of Mathematics, Sun Yat-Sen University, Guangzhou 510275, China
E-mail: yanx39@mail2.sysu.edu.cn

2. Business School, Sun Yat-Sen University, Guangzhou 510275, China
E-mail: xiali5@sysu.edu.cn (corresponding author)

3. Department of Automation, Tsinghua University, Beijing 100084, China
E-mail: {zhaoqc, yangjun603}@tsinghua.edu.cn

Abstract: Texas Hold'em poker is a popular game worldwide and it attracts increasing attention from community of artificial intelligence as a typical decision-making problem in non-deterministic and incomplete information environment. One of the key tasks to deal with the poker game is the opponent modeling, which aims to exploit the opponent weakness based on history behaviors. In this paper, we study mixed-method opponent modeling, one is Bayesian probabilistic model, one is the neural network (NN)-based prediction model, and the last is opponent type identifying model. Then, we combine these three methods to generate an integrated agent for opponent modeling. The opponents are categorized into 4 types according to their risk preference of strategies. The main step in Bayesian method is calculating the posterior distribution over opponent's strategy space and selecting the maximum probability. The main idea of NN-based method is using observation data to improve the prediction accuracy of opponent's hand. The main idea of opponent type identifying model is building a classifier with two factors. Finally, we design a simplified poker game to conduct experiment and demonstrate the effectiveness of our methods.

Key Words: Opponent modeling, Bayes, Neural network, Opponent type, Enumeration

1 Introduction

The game of poker presents a serious challenge to artificial intelligence since poker is a typical game with imperfect information, which means we do not know our opponents private card (also known as hand), and the game environment is non-stationary. Unlike Chess or Go, we do not know the exact states of the whole system in the poker game since we do not know the next state our action will lead to and what kind cards will be dealt by shuffle deck. These uncertainties make poker game a challenge problem.

The method of solving poker problem has been developed many years, many effective poker AI have been created[1–3]. Among these implemented methods, the most common way is using game theory-based approach to find a equilibrium strategy, to be specific is by approximating Nash equilibrium. Until now, the most popular way is using CFR[4] or combining neural network with reinforcement learning like NFSP[5]. Unlike equilibrium strategy method, opponent modeling method views every opponent differently and views our opponent as a player who has weakness to be exploited.

There are two main reasons why we need opponent modeling in poker games. First, equilibrium strategy methods are on the basis of the assumption that opponents are quite smart without weakness. From the strategic perspective, it is the optimal strategy we can find when our opponent also acts with optimal strategy. But from the exploratory perspective, they represent a pessimistic viewpoint. If our opponents are not as smart as expected, we can model on our opponents to obtain higher payoffs than the Nash value of the game. Another importance of opponent modeling is serving as a key part of other methods. For example, we can predict opponent hand to convert imperfect information games into perfect information games. Also a world-class poker agent often be the combination of opponent modeling and other methods. So it is obvious that opponent modeling is an important part in poker games.

The research of opponent modeling has a long history, it is thought to be the combination of exploration and exploitation. Many effective methods have been raised, such as predicting opponent hand[6] base on historical action. Predicting opponent action[7] with two important factors: historical actions and amounts of previous calls. Identifying opponent type[8] base on two factors. One Bayesian method is by calculating a Bayesian best response to our opponent[9]. Opponent modeling can also be tracking opponent changing[10]in the game to help us change our strategy. Besides, we can compute an opponent model by identifying opponent deviation between a precomputed equilibrium strategy and actions in the game[11]. If we have enough experts to guide our agent, then we can choose different expert to achieve short-term exploitation[12]. In recent years, a more effective method of opponent modeling is raised, which uses LSTM neural network to exploit opponent base on action sequences in computer poker[13, 14]. In summary, there are many research achievements in this domain.

In this paper, we will present a mixed-method model of opponent modeling including Bayesian method[9], neural network (NN)-based method[7, 15], enumeration method[6], opponent type identifying method[8], and strategy evaluation method[12]. In our model, these methods are combined together to create an agent which can play complete poker game and highly explore opponent weaknesses. We use enumeration method to play the game at the beginning. As the game progresses, we identify opponent type to help us choose response and we use NN-based method as a short-term exploring method which models on opponent character. We calculate a bluff response by Bayesian method which models on opponent strategy. But it is not enough for only thinking about how to exploit our opponent. Comparing with equilibrium strategy, opponent modeling is not a safe

This work is supported by National Natural Science Foundation (NNSF) of China under Grant 00000000.

978-1-7281-5922-5/20/$31.00 ©2020 IEEE

method, so we need strategy evaluation method.

Our model aims to give a complete opponent modeling-based strategy in computer poker. Often an opponent modeling method is aiming at solving a part of problems in poker games such as how to predict opponent action, or only concerning about how to play the game after we have enough observations. These methods are meaningful but unlike CFR method, it dose not give a strategy which can be used in the complete competition against an opponent.

Texas Hold'em often viewed as the testbed to test method of solving poker problem. It is a typical extensive game which has enough uncertain elements such as the deck. But Texas Hold'em has a quite large game tree and it requires abstraction. Considering this, we need to implement our model on a reduced Texas Hold'em. Here we design a small poker game *Extended Leduc* which is similar to Texas Hold'em poker and more complicated than Leduc poker[16].

2 Problem Description

In this section we introduce our poker games first.

2.1 Extended Leduc Poker Game

There are many types of poker games, in this paper we focus on limited heads up hold'em (i.e, two players with prespecified bet, call and raise amounts). We construct a small version of hold'em, which seeks to retain the strategic elements of the large game while we do not need to do abstraction.

In our poker game, there are two players and three rounds at all, the deck consists of two suits with six cards in each suit (89TJQK, no color). Before the game starts, both players start the first round with 1 (one chip) already in the pot. Each round started with P1, each player makes decisions only one time at each round. We denote Player 1 as P1 and player 2 as P2, and our agent is P1.

Round

- At the first round, a card is dealt to each player as hand. P1 can either bet with amount 2 or check with no amount added. After P1 takes an action. If P1 bets, P2 can either call with amount 2 or raise with amount 4. If P1 checks, P2 can either bet with amount 2 or check with no amount added.
- At the second round one public card is revealed. P1 can either bet with amount 4 or check with no amount added. After P1 takes an action. If P1 bets, P2 can call with amount 4, raise with amount 8 or fold. If P1 checks, P2 can either bet with amount 4 or check with no amount added.
- At the third round, if no one folds, one public card is revealed. P1 can either bet with amount 6 or check with no amount added. After P1 takes an action. If P1 bets, P2 can call with amount 6, raise with amount 12 or fold. If P1 checks, P2 can either bet with amount 6 or check with no amount added.

If there only remain one player in the game, then the player wins the pot. Otherwise, both players' hand are showed on the table. Each player¡¯s hand is combined with board (all revealed public cards) as combined cards for comparing, winner is judged by the card strength of combined cards.

Card Strength

- Straight: three cards in order (e.g. JQK).
- One pair: at least two cards in a pair (e.g. JJQ).
- High card: other combinations (e.g. TQK).

Card strength is listed in descending order. If two combined cards have same card strength, then comparing the largest single card. If still no one wins, this hand end in a draw, players split the pot.

2.2 Hand

Since we establish our model base on observations, we need to define hand information to describe hands with perfect information. We use H to denote one hand.

Hand Information Definition

- C: P1's hand.
- D: P2's hand.
- $R_{1:k}$: R_i is the revealed public card at round i.
- $A_{1:k}$:A_i is P1's decision at round i.
- $B_{1:k}$: B_i is P2's decision at round i.
- K: Kind of hand, showdown or foldcard.

Then $H = (C, D, R_{1:k}, A_{1:k}, B_{1:k}, K)$

2.3 Information Set

Information set is often used in extensive games, which in game theory means sets of game states that players cannot distinguish themselves. In such states they choose actions with the same distribution. Here we use information set to describe our state in the game, and we denote information set with I

Information Set Definition

- Hand: a player's private card.
- Action sequence h: historical action sequence of all players so far which presents in order.
- P_h: next player to take action (e.g. if action sequence is "bet", then the next player to take the action is P2).
- S: action space of P_h (e.g. bet or check).
- R: public cards revealed so far in the game.

2.4 Strategy

A behaviour strategy specifies a distribution over actions space S for every information set of player i. For example, if an information set has the following action space: bet or check. Then the strategy of this information set is probability $p1$ and $p2$ ($p1 + p2 = 1$), which means at this information set the player choose to bet with probability $p1$ and choose to check with probability $p2$.

2.5 Problem

Except challenges mentioned above. In our poker, if opponent folds, then we do not know his hand. Other challenges such as the high variance of payoffs, also known as luck, it is also a factor that cannot be ignored.

Our problem is when against an opponent who has weakness, considering all challenges mentioned above and under some necessary assumptions, how to win high payoff by opponent modeling method no matter what type it is. Assuming that against an opponent we play K hands and our payoff at each hand is V_i. Our problem can be described as how to find a strategy ψ which:

$$max \sum_{i=1}^{K} V_i(\psi | assumptions) \quad (1)$$

2.6 Method

Since many opponent modeling methods are effective in particular situations , we can build our model by combining these methods together and give a complete strategy which uses suitable response in different situations.

Our model consists of a classifier based on opponent type identifying method and two learner, one is Bayesian learner based on Bayesian method and another one is Neural Network learner based on Neural Network method. Besides, considering that there are hands we cannot use opponent modeling method and opponent modeling is dangerous. We complete our model by adding an initial strategy based on enumeration method and a switchboard based on evaluation method. This model gives our agent a complete strategy to win high payoffs in *Extended Leduc* poker game.

2.7 Statement

Two statements must be made. First, opponent modeling cannot guarantee expected payoff. We might calculate a high expected payoff in our strategy but sometimes our real payoff is unsatisfactory. Second, opponent modeling is effective while our opponent is stable for some times. It is quite difficult for exploiting when against an ever-evolving opponent.

We will introduce all methods in our model and then give complete strategy in following sections.

3 Opponent Type Classifier

This section we introduce a classifier based on opponent type identifying method. We need to identify opponent type because different opponent has different character, then we can have different response. In professional Texas Hold'em, opponents are often divided into four types. Since our poker game is reduced Texas Hold 'em, we use this classification in our model.

Opponent Type

- TightAggressive: this type of opponents are good at playing. They exactly know how to protect their hand and they know clearly whether fold or raise. So it is hard to exploit this type of opponent.
- TightPassive: this type of opponents often move according to their hand, so their actions often mean what kind of hand they hold. We can explore them from their historical actions.
- LooseAggressive: this type of opponents tend to be aggressive. They often bet or raise to put pressure on their opponents. The best choice against this type of opponents is losing some small pots but win a big one.
- LoosePassive: this type of opponents tend to be conservative. They usually call even it is the time to raise or fold and highly depend on luck.

Our classifier works according to two factors: AF and $VPIP$, where

$$AF = \frac{NumRaise}{NumCalls} \tag{2}$$

$$VPIP = \frac{Num\ of\ Raise\ at\ least\ once}{NumHands} \tag{3}$$

Then we have table 1.

Table 1: Classifier

	$VPIP < 0.28$	$VPIP \geq 0.28$
$AF \geq 1$	TightAggressive	LooseAggressive
$AF < 1$	TightPassive	LoosePassive

4 Neural Network Learner

This section we introduce a Neural Network learner based on Neural Network. This learner uses NN-based method to predict opponent hand by observations collected in the first two rounds and it only give strategy at the third round.

There are many factors can affect a player's decision in poker games, such as the strength of the combination of hand and revealed public cards, other players' actions, size of the pot and so on. So it is possible for us to model on opponent character based on observations and even in short-term observations we can still identify some useful characters. Considering that luck is also an important element which can affect a player's decision, we need to use more observations to reduce its influence. Next we show how to use a feed-forward neural network to predict opponent hand.

4.1 Predict Opponent Hand

Background

There is a classical way to predict opponent hand by using statistic method[6] and it is implemented on Texas Hold'em. Before game starts, an initial *weight table* is maintained for each opponent. After each round, the table is updated according to opponent action. Finally the table is the probability of each kind of hand the opponent might hold.

Now, we can use a feed-forward neural network to approximating the function used for updating the table.

It is difficult to predict a specific hand of opponent because opponent might choose same action for those hands that are strategically similar. We need to define a card level evaluation system for two cards (hand and revealed public card) which takes current card strength and potential into consideration. We do not predict specific hand but the level of combined cards.

Network

Network used for hand predicting learner is a two-layer feed-forward neural network. There are 20 neurons in input layer, 9 neurons in the hidden layer, and 3 neurons in output neurons (each representing a level of combined cards). We use dummy variable to deal with the input variable which is a common practice in statistics. We use LM algorithm to train this neural network.

Input

- P1's actions at the first and second round.
- P2's actions at the first and second round.
- Revealed public card at the second round.

Target

- Level of combined cards.

4.2 Reweight Hand Strength

Assuming that Neural Network has been trained. At the third round, another public card is revealed. We can obtain predicted level, then predicted hands can be calculated by predicted level, board, and our hand. We denote the combination of P1's hand and board as C^1, combinations of P2's

predicted hands and board as $C_1^2 \cdots C_i^2$, and combinations of P2's rest possible hands and board as $C_{i+1}^2 \cdots C_k^2$. We can calculate a reweight hand strength RHS with predicting accuracy A by:

$$RHS = \frac{A \cdot \sum_{j=1}^{i} \tau(C_j^2)}{i} + \frac{(1-A) \cdot \sum_{j=i+1}^{k} \tau(C_j^2)}{k-i} \tag{4}$$

$$\tau(C_j^2) = \begin{cases} 1, & \text{if } C_j^2 \text{ can win } C^1 \\ \frac{1}{2}, & \text{if } C_j^2 \text{ is tied with } C^1 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Neural Network learner is trained base on observations, and calculates RHS at the third round after training. Our agent moves at the third round according to calculated RHS.

5 Bayesian Learner

This section we introduce Bayesian leaner based on Bayes formula. This learner uses Bayesian method to calculate a response which is good at bluffing and defines at every information set. We denote P1's strategy by α and P2's by β. Before introducing, we give some assumptions first.

5.1 Assumption

First, P2's strategy is stationary. Second, players' strategies are independent. More formally, $P(\alpha, \beta) = P(\alpha) \cdot P(\alpha)$. Finally, we assume the deck is uniformly distributed. These assumptions imply that all hands are i.i.d.given the strategies of the players.

5.2 Probability of Hand Occurring

Suppose a hand has been observed, we can calculate the probability of it occurring given opponent strategy.

Showdown hand H_s

$$\begin{aligned} P(H_s|\beta) &= P(C, D, R_{1:k}, A_{1:k}, B_{1:k}|\beta) \\ &= P(C)P(C|D) \prod_{i=1}^{k} [P(B_i|D, R_{1:i}, A_{1:i}, B_{1:i-1}, \beta) \\ &\qquad P(A_i|C, R_{1:i}, A_{1:i-1}, B_{1:i-1}) \\ &\qquad P(R_i|C, D, R_{1:i-1})] \\ &= P(C)P(C|D) \prod_{i=1}^{k} [\alpha_{I_{1i}, A_i} \beta_{I_{2i}, B_i} \\ &\qquad P(R_i|C, D, R_{1:i-1})] \\ &= p_{\text{showdown}} \prod_{i=1}^{k} \alpha_{I_{1i}, A_i} \beta_{I_{2i}, B_i} \end{aligned} \tag{6}$$

α_{I_{1i}, A_i} is the probability of P1 take the action A_i at information set I_{1i} which consists of $C, R_{1:i}, A_{1:i-1}$ and $B_{1:i-1}$, dictated by P1's strategy α. β_{I_{2i}, B_i} is the probability of P2 take the action B_i at information set I_{2i} which consists of $D, R_{1:i}, A_{1:i}$ and $B_{1:i-1}$, dictated by P2's strategy β. Because of assumption, $P(C), P(C|D), P(R_{1:i-1})$ only depend on the number of cards dealt to each player. We replace these product with p_{showdown}. Considering that P1's strategy would not change when opponent's strategy is different, so the final unnormalized probability depends only on β.

Foldcard hand H_f

$$\begin{aligned} P(H_f|\beta) &= P(C, R_{1:k}, A_{1:k}, B_{1:k}|\beta) \\ &= P(C) \sum_{D} P(C|D) \\ &\quad \times \prod_{i=1}^{k} [P(B_i|D, R_{1:i}, A_{1:i}, B_{1:i-1}, \beta) \\ &\quad P(A_i|C, R_{1:i}, A_{1:i-1}, B_{1:i-1}) \\ &\quad P(R_i|C, D, R_{1:i-1})] \\ &= P(C) \sum_{D} P(C|D) \prod_{i=1}^{k} [\alpha_{I_{1i}, A_i} \beta_{I_{2i}, B_i} \\ &\quad P(R_i|C, D, R_{1:i-1})] \\ &= p_{\text{foldcard}} \prod_{i=1}^{k} \alpha_{I_{1i}, A_i} \sum_{D'} \prod_{i=1}^{k} \beta_{I_{2i}, B_i} \end{aligned} \tag{7}$$

In a foldcard hand we do not observe opponent hand, so we need to list all possible opponent hand D'. Similar with showdown hand, p_{foldcard} is a function that depends only on the number of cards dealt to the each player and the number of public cards revealed so far. So the final unnormalized probability depends on β.

5.3 Posterior Distribution

Given a set $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_f$ of observations, where $\mathcal{O}_s$ are observations of hands that led to showdown and $\mathcal{O}_s$ are observations of hands that led to foldcard. We can compute the posterior probability distribution over the space of opponent strategies. Using a simple Bayes' rule, we can obtain the following formula:

$$\begin{aligned} P(\beta|\mathcal{O}) &= \frac{P(\mathcal{O}|\beta)P(\beta)}{P(\mathcal{O})} \\ &= \frac{P(\beta)}{P(\mathcal{O})} \prod_{H_s \in \mathcal{O}_s} P(H_s|\beta) \prod_{H_f \in \mathcal{O}_f} P(H_f|\beta) \\ &\propto P(\beta) \prod_{H_s \in \mathcal{O}_s} P(H_s|\beta) \prod_{H_f \in \mathcal{O}_f} P(H_f|\beta) \end{aligned} \tag{8}$$

5.4 Response

When we can use observations to obtain the posterior distribution over opponent strategy space, we need to think how to calculate a response to our opponent.

Bayesina Best Response

We start with how to obtain Bayesian best response (*BBR*), it is a response no matter what opponent strategy is that maximize the expected value we can win. We have following formula:

$$\begin{aligned} \alpha^{\text{BBR}} &= \underset{\alpha}{\text{argmax}}\, \text{E}_{\mathcal{O}|\text{H}} \text{V}(\text{H}) \\ &= \underset{\alpha}{\text{argmax}} \sum_{\text{H} \in \mathcal{H}} \text{V}(\text{H}) \int_{\beta} P(\text{H}|\alpha, \beta, \mathcal{O}) P(\beta|\mathcal{O}) \end{aligned} \tag{9}$$

Next we consider how to make this formula can be implemented. Formula (9) has been proved can be calculated by constructing a special game tree and in this paper we call it *Expectimax tree*[9].

Expectimax Tree

First we denote the node information of the tree, there are four types of nodes in the tree.

- Chance node: it is a node for dealing hand to each player or revealing public cards.
- P1 action node: it is a node for P1 to take an action over action space, corresponds to an information set.
- P2 action node: it is the node for P2 to take an action over action space, corresponds to an information set.
- Leaf node: it is the enumeration of all P2's possible cards.

Then we construct and update this tree by adding all possible observations in order they would be observed by P1, including P1's hand, action sequence, revealed public cards, and P2's possible cards. For a leaf node the value should be the payoff to P1 multiplied by the probability of P2's actions reaching this leaf given the posterior distribution over strategies.

For a P2 action node or a chance node, the value is the sum of its children values. For a P1 action node, the value is the maximum of its children values, and BBR assigns probability one to the action that leads to the maximal child over this node¡¯s corresponding information set.

Repeating adding observation and updating the node value until every node has been assigned a value, then we obtain BBR. The root value can be written as:

$$\sum_{R_1} \underset{A_1}{max} \sum_{B_1} \cdots \sum_{R_k} \underset{A_k}{max} \sum_{B_k} \sum_{D} \int_{\beta} \prod_{i=1}^{k} \beta_{I_{2i},B_i} \boldsymbol{P}(\mathcal{O}|\beta)\boldsymbol{P}(\beta) \tag{10}$$

MAP Response

Computing the integral in formula (10) over opponent strategy space is difficult, so we need to figure out some simple but effectively ways to make this computation easier. Here we introduce MAP response.

MAP means max a posterior. We find the strategy which has the maximal posterior probability over opponent calculated posterior distribution. In practice, we sample a set of strategies from priors, and calculate their posterior probability by formula (8). Using the one which has maximum probability as opponent strategy, then find the BBR to that strategy.

This approach is straightforward and crude. Although it may never fully explore opponent strategy, it requires much less computing resource and its effect is still obvious.

In our poker game, the number of information sets is not big, so we use Dirichlet distribution to specifies a distribution over action space for every information set. Dirichlet (2, 2, 2) distribution is used for fold, call, and raise. Dirichlet(2, 2) distribution for bet and check.

Bayesian learner is trained base on observations of both showdown and foldcard hands then MAP response is given.

6 Complete Strategy

This section we introduce two methods to complete our model first, then we give the complete strategy ψ in our model.

6.1 Initial Strategy

First we introduce initial strategy based on enumeration method. At the beginning of the game we do not know too much about our opponent so an effective strategy to play the game is making decisions according to our hand strength.

First Round

At the first round we only know our our own hand, so we use Monte Carlo drop. For each hand, we sample 100 opponents and against each opponent 100 hands is simulated with P1 only bet at the first round and 100 hands with P1 only check at the first round. Incomes are divided by total payoffs and stored in a 6×2 $income\ rate\ table$. Our agent checks the table and takes the action with higher value.

Second Round

At the second round, one public card is revealed, so we make decisions according to our hand and revealed public card. First we calculate current hand strength by enumerating all possible opponent hands and rest public cards, then recording the number of times we win the game as P, end in a draw as T and lose the game as N. Current hand strength CHS is calculated by:

$$CHS = \frac{P + T/2}{P + T + N} \tag{11}$$

Since there is still one public card not revealed, we need to consider the potential of our hand. Here we record the number of times our CHS is weaker than opponent but win the game as NP and end in a draw as NT, CHS is stronger than opponent but lose the game as PN and end in a draw as PT, CHS is same as opponent but win the game as TP and lose the game as TN. Also we record the number of times our current hand strength is stronger than opponent as CP, weaker than opponent as CN and same as opponent as CT. $Ppot$ is calculated by:

$$Ppot = \frac{NP + NT/2 + TP/2}{CN + CT} \tag{12}$$

$Npot$ is calculated by:

$$Npot = \frac{PN + PT/2 + TN/2}{CP + CT} \tag{13}$$

Then effective hand strength EHS is calculated by:

$$EHS = CHS \cdot Npot + (1 - CHS) \cdot Ppot \tag{14}$$

Before calculating EHS, CHS need to be normalized. Our agents bets if $EHS_{\text{hand,board}} > pot\ odds$ ($pot\ odds$ is the ratio of the current betting amount to pots).

Third Round

At the Third round, since board are all revealed, it is easier to calculate our hand strength since the only hidden information is opponent hand. Hand strength HS is calculated same as CHS. Our agent bets if $HS_{\text{hand,board}} > pot\ odds$.

6.2 Switchboard

Since Bayesian method only promise high expected payoff and sometimes strategies sampled from prior is far from the real strategy. So MAP response sometimes might not work well. Also our strategy might be explored by our opponent and be exploited in reverse. So we need to build a

switchboard to decide whether switch to a conservative strategy by evaluating our current strategy.

The most straightforward way to evaluate our strategy is the payoffs, also called as $score$. Switching does not depend on opponents because sometimes our opponent might change strategy but our strategy still work well, in this situation we do not need to change strategy.

Here the switchboard calculates the score of current strategy not just depend on winning pots but each hand's score by:

$$score = \frac{s_8}{6 \cdot c_8} + \cdots + \frac{s_K}{6 \cdot c_K} \tag{15}$$

where s_{card} is the total winnings pots when our hand is $card$, c_{card} is the number of times when our hand is $card$.

6.3 Strategy ψ

We divided the process against an opponent into three stages. At first stage our agent uses initial strategy to play the game.

At second stage our agent can explore opponent character base on some observations. First our agent identifies opponent type by classifier. If opponent type is TightPassive or LoosePassive, our agent uses observations of the first stage to train Neural Network learner and at the third round obtain RHS. If $RHS > potodds$, bets, or checks. If opponent type is LooseAggressive, our agent keeps checking unless our hand is a good one. If opponent type is TightAggressive, then our agent still use initial strategy because this type of opponent is hard to be exploited. We call the strategy used in this stage as NN response.

At third stage our agent uses observations of first two stages to train Bayesian learner and obtain MAP response. There are two differences compared with pure MAP response. The first one is that our agent uses NN response at third round if opponent type is TightPassive. In other situations our agent uses MAP response. The second difference is that the switchboard evaluates current strategy every h hands. If $score < threshould$, then switchboard turns on and switches to NN response until our agent obtains a new MAP response. We do this to increase the security of opponent modeling.

7 Experimental

7.1 Experimental Setup

We establish our experiment on 100 trials. For each trial one opponent is sampled from prior. Except our agent, we create other two high level agents for comparison. They both move according to their hand. The main difference is that one is more conservative and the other one is more adventurous. Three agents all play 1000 hands against sampled opponent each trial. We record *accumulated average bankroll* (accumulated average winning amount per trial), *winning rate* and *bluff rate* (ratio of foldcard hands by opponent to all hands) at each trial.

About our agent, we denote the first stage is 100 hands and the second stage is 100 hands. We use 200 hands to obtain MAP response and evaluate current strategy every 200 hands. For Bayesian learner, 10000 hands are generated randomly before the game to obtain all possible observations that P1 might encountered. For each trial, 1500 strategies are sampled from priors as opponent strategy space. For Neural Network learner, we create card level evaluation system based on EHS. Combined cards at the second round are evenly divided into three levels according to EHS.

Since opponents sampled from prior is different from human player, we created another 20 human-like opponent to test Neural Network learner. Against each opponent 200 hands are played and Neural Network is trained base on 200 hands, 150 hands as learning set and 50 hands as test set. Here we have 1000 predictions, we show prediction accuracy by quartile and prediction error by confusion matrix of 1000 predictions.

To demonstrate the validity of our model, we plot the accumulated average bankroll, histogram of winning rate and histogram of bluff rate of our agent, conservative agent and adventurous agent.

7.2 Results

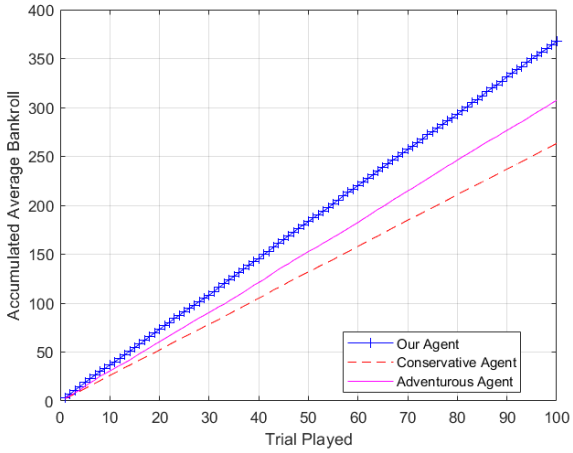

Fig. 1: Accumulated average bankroll of three agents

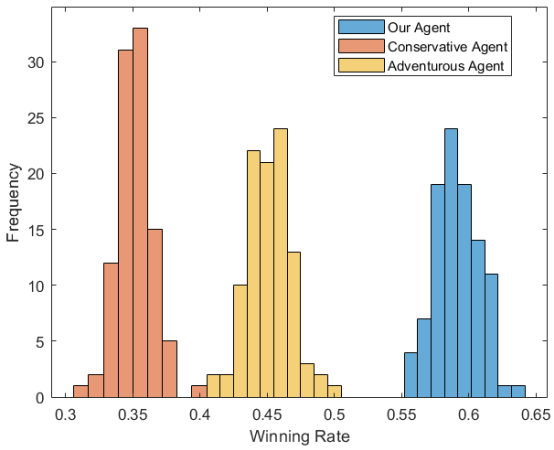

Fig. 2: Histogram of three agents' winning rate

Among 100 trials, average maximum posterior probability is 83.11%, which tell why we use MAP response. We do not need to consider the rest sample strategies while MAP strategy has probability higher than 80%. It is clear to see that our agent behaves well in 100 trials (Fig. 1, Fig. 2, Fig. 3).

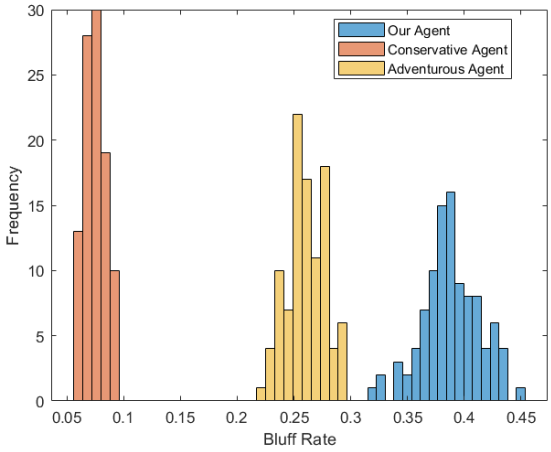

Fig. 3: Histogram of three agents' bluff rate

For winning amount, our agent can win 3.7 each hand on average while adventurous agent win 3.0 and conservative agent win 2.6, it is thought to be high payoff in our poker game. For winning rate, average winning rate of our agent is 60%, while adventurous agent is 44% and conservative agent is 35%. It is satisfying too. For bluff rate, average bluff rate of our agent is 38.86%, while adventurous agent is 26.03% and conservative agent is 7.47%. Winning rate and bluff rate show that our agent wins more hands on the basis of forcing our opponent to fold by effectively modeling on opponent weakness. But compare with other two agents, our agent's payoff dose not improve significantly. It is because our agent tend to be more aggressive, in many showdown situations it chooses an action with low probability but higher expected value while other two agents chooses an action that matches their hand strength. Although our agent wins more pots by opponent folding, it loses more pots because of taking dangerous actions. And in 100 trials, switchboard helps change 52% poor MAP response to a good one, it is a main reason for dangerous MAP response to be sustainable.

For the test of Neural Network learner, the first quartile of prediction accuracy is 64%, the second one is 66% and the third one is 68%. Average accuracy is 65.70%. Only considering overall prediction accuracy is not not enough. From Confusion Matrix (Table 2) we can see that Neural Network has about 65% prediction in predicting each level of combined cards. It is not a high number but balance, and it is higher than weight table method. So Neural Network learner can though to be effective.

Our agent is more like an aggressive player, might loose more pots than considerate player in some situations, but win more pots by exploiting opponent weakness. We can say that our agent can effectively win high payoffs against most types of opponents in our poker game.

Table 2: Confusion Matrix

		Predicted		
		level 1	level 2	level 3
Actual	level 1	66.45%	22.80%	10.75%
	level 2	17.03%	63.19%	19.78%
	level 3	9.42%	22.80%	67.78%

8 Conclusions

From experiment result we found that opponent modeling can win higher payoffs in poker games. It can effectively exploit opponent weakness but more dangerous. A main reason why opponent modeling is not as popular as equilibrium strategy method is that opponent modeling requires too many assumptions and some of them are unrealistic. But with the development of opponent modeling, these disadvantages can be removed by combining other opponent methods together or more advanced technology. We believe the building process of an excellent poker AI can not leave opponent modeling and there is a lot of space for exploration in this field.

Our future work is to study how to apply these methods to a more complex dynamic environment, how to explore non-stationary opponents and how to make opponent modeling more safe while remain exploratory.

References

[1] Brown, Noam, and Tuomas Sandholm. "Superhuman AI for multiplayer poker." Science 365.6456 (2019): 885-890.

[2] Moravcik, M. et al. Deepstack: expert-level artificial intelligence in heads-up nolimit poker. *Science*,356, 508¨C513 (2017).

[3] Bowling,Burch, Johanson,Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347, 145-149(2015)

[4] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. (2007). Regret minimization in games with incomplete information. In Advances in Neural Information Processing Systems, pages 1729¨C1736.

[5] Heinrich, Johannes, and David Silver. "Deep reinforcement learning from self-play in imperfect-information games." arXiv preprint arXiv:1603.01121 (2016).

[6] D. Billings, D. Papp, J. Schaeer, and D. Szafron. Opponent modeling in poker.In *AAAI National Conference*, pages 493,p499, 1998.

[7] Davidson, Aaron, et al. "Improved opponent modeling in poker." International Conference on Artificial Intelligence, ICAI¡¯00. 2000.

[8] Te¨®filo, Lu¨ªs Filipe, et al. "Adapting strategies to opponent models in incomplete information games: a reinforcement learning approach for poker." International Conference on Autonomous and Intelligent Systems. Springer, Berlin, Heidelberg, 2012.

[9] Finnegan Southey,Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, Chris Rayner,Bayes' Bluff: Opponent Modelling in Poker,Appears in*Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI2005)*,4 Jul 2012.

[10] Everett, Richard, and Stephen Roberts. "Learning against non-stationary agents with opponent modelling and deep reinforcement learning." 2018 AAAI Spring Symposium Series. 2018.

[11] Ganzfried, Sam, and Tuomas Sandholm. "Game theory-based opponent modeling in large imperfect-information games." The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[12] Hoehn, Bret, et al. "Effective short-term opponent exploitation in simplified poker." AAAI. Vol. 5. 2005.

[13] Li, Xun, and Risto Miikkulainen. "Evolving Adaptive LSTM Poker Players for Effective Opponent Exploitation." (2017).

[14] Li, Xun, and Risto Miikkulainen. "Dynamic adaptation and opponent exploitation in computer poker." Workshops at the

Thirty-Second AAAI Conference on Artificial Intelligence. 2018.
[15] Bhat, Gautam, Kaviarasan Selvam, and Veena Dali. "NN-based Poker Hand Classification and Game Playing."
[16] Waugh, Kevin, et al. "Abstraction pathologies in extensive games." Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 2009.