Poker Project - Team #26

Jia Yilin, Liow Zhu Hui, Na Yi Rong, Phee Hay Bryan, Yoon Jia Jun Ken
National University of Singapore
{e0007663, e0176696, e0127341, e0014997, e0031814} @u.nus.edu

Abstract

In this project, we design an intelligent agent to play Limit Texas Hold'Em Poker. We propose a novel adaptive agent designed to predict an accurate representation of its hand strength, through Monte Carlo simulations, and to extract information about the opponent's behaviour and hand strength through opponent mapping. The agent utilises a nested evaluation function to determine its actions through Bayesian Inference, adapting its strategy based on the opponent.

1 Introduction

Artificial intelligence (AI) has advanced significantly in the 21st Century, and have come to play a critical role in improving efficiency and efficacy in many industries [West and Allen, 2018]. A big challenge in AI is the lack of complete information in the real world, hence designing an AI agent to work in an imperfect information environment is critical. The game of poker embodies such an environment, and we tackle this project by designing an AI agent for poker.

This project will focus its efforts on Heads Up Limit Texas Hold'Em Poker, a two-player game of the poker variant. Small and big blinds are fixed at 10 and 20 dollars respectively with the amount of raise value is limited to 10 dollars per raise, up to a maximum of four raises per round. The rounds include pre-flop, flop, turn and river. Two players, the proposed agent and the opponent, face off against each other in a tournament of 500 games, each starting with 10,000 dollars. This presents a suitable framework of the game for us to simulate the performance of our agent.

Conventional poker agents compute an optimal strategy by approximating a Nash Equilibrium using a minimax tree search through nodes of possible hands and actions, and running an evaluation function to determine the utility of each node. The most logical criteria to consider is the strength of the player's hand at each node, determined by evaluating the possible future game states. Due to the stochastic nature of poker, the branching factor of such a tree is extensive and results in astronomical search space and search time. In this report, we aim to do the following:

1. Illustrate the viability and effectiveness of Monte Carlo simulations for hand strength calculation
2. Analyse the opponent's hand strength through a preliminary evaluation function (EF)
3. Nest the aforementioned EF into a final EF that computes overall win rate given both players' hand strengths

The final EF will be trained through Reinforcement Learning to determine the threshold values that govern each action.

2 Motivation

2.1 Monte Carlo Simulations

It is possible to generate a heuristic that determines the current hand strength without generating a game tree. Ultimately, the minimax search returns an optimal strategy based on the probability of winning the current game. Instead of calculating expected value of winning through the game tree, we can do so by running multiple simulations of games with the current hole cards using the function "estimate_hole_card_win_rate". The formula used for this calculation is shown below for n games where o_i has a value of 1 when the game is won, and a value of 0 otherwise.

$$WinRate = \frac{\sum_{i=1}^{n} o_i}{n} \quad (1)$$

Winning m number of games in n simulations where the player never folds, the chance of winning given a specific hand can be estimated to be *100% * m/n*. This is the probability that the player will win given the current hole cards and community cards. This Monte Carlo approach provides an accurate estimate of the win rate when the value of n is large due to the statistical law of large numbers [Routledge, 2005]. However, the time limitation of 200ms per round to decide our agent's action prevented us from running too many simulations. The number of simulations used to calculate the heuristic had to be selected carefully.

Using the truth values of winning odds given only 2 hole cards [Shackleford, 2017], we ran simulations for n=100 and n=1000 to determine the difference in accuracy. For each hand, the average was taken over 1000 different pairs of hole cards. The results are as shown in Table 1.

Table 1: Performance Comparison for Different Number of Monte Carlo Simulations

n	Average Deviation from True Odds	Largest Deviation from True Odds
100	2.46%	6.10%
1000	2.43%	4.15%

Although the largest deviation observed was significantly lower when n=1000, using the larger value of n was negligibly better when it came to average deviation. This was in spite of the simulations taking 10 times the amount of time taken when n=100. Further tests were run to ensure the accuracy of Monte Carlo simulations when more community cards are known. The results are shown in Table 2.

Table 2: Performance Comparison for Different Number of Community Cards

Number of Open Community Cards	Average Deviation
3	0.892%
4	0.856%
5	0.833%

The results in Table 2 confirm our belief that setting n to be 100 gives a good compromise and provides appropriately accurate results while using little time.

2.2 Opponent Behaviour

Taking into consideration that the opponent plays logically to maximise its winnings, the opponent's action will generally be based on its hand strength, raising with a strong hand and calling or folding on weaker hands. It can be assumed that its level of aggression is proportional to its hand strength. Our agent exploits this information and estimates the opponent's strategy from the association between the opponent's current and past behaviours. Against any new opponent, our agent will adopt a minimum bet strategy, where it will neither fold or raise for the initial 25 hands to learn about the opponent's playstyle and its raise threshold at each round. In doing so, the agent maximises information acquisition by revealing the opponent's hand at the end of each game.

Additionally, our agent adapts to our opponents' playstyle and adjust its strategy against different players to maximise its output accordingly. In the poker terminology, the type of players can be classified into loose passive, tight passive, loose aggressive and tight aggressive. Our agent exploits and pressures against tight-passive players, outsmarting tight-passive and tight-aggressive players with their known thresholds and place emphasis on its own hand strength against loose-aggressive players.

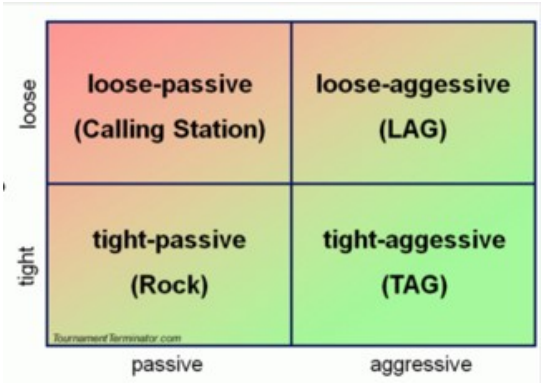

Figure 1: Classification of Poker Players [TournamentTerminator.com, 2010]

Due to the scope of this project, our agent will only be competing against the other agents. However, our agent is also designed to compete reasonably well against other human players as it is able to learn their playstyle and exploit their hand ranges.

3 Agent Design

3.1 Heuristics

For the final design of our agent, a total of four heuristics (h_i) were considered and implemented. They are further defined in the following subsections:

h_1 - Immediate Hand Strength

Immediate Hand Strength (h_1) is defined as the statistical win rate obtained from the inbuilt "estimate_hole_card_win_rate" function as discussed in Section 2.1. This method uses Monte Carlo simulations to estimate the win rate given the current open cards.

h_2 - Historical Opponent Behaviour

Historical Opponent Behaviour (h_2) classifies the opponent according to his behaviour during the game. Players are classified as either aggressive, passive, or logical. Different strategies are taken with each type of player, as shown previously in Section 2.2. This allows us to exploit their behaviour to improve our chances of winning.

This heuristic is calculated using the opponent's past action history, as well as his hand strength during the various prior rounds. A player is considered to be illogical and aggressive if a substantial proportion of his past actions were raises rather than calls or folds, and he loses the round with a weak hand. Conversely, a player is considered as illogical and passive if he rarely raises his bet even though his hand was strong. This level of aggressiveness will affect the raise multiplier (mRaise) and call multiplier (mCall), which will be utilised below.

The values of the two multipliers will be dependent on opponent behaviour. mRaise ranges from 1.0-1.7 and mCall ranges from 0.8 - 1. The opponent behaviour, h_2, ranges from 0-1, with 1 indicating the highest passiveness. The two multipliers will be calculated as follows:

$$mRaise = 1.05 + (1.7 - 1.05)h_2 \quad (2)$$

$$mCall = 0.4 + (0.95 - 0.4)h_2 \quad (3)$$

The values were experimentally determined and are logical, as a raise by a passive opponent generally indicates an abnormally strong hand, therefore the mRaise should be higher.

h_3 - Opponent Raise Threshold

Opponent Raise Threshold (h_3) keeps track of the hand strength threshold above which the opponent will raise his bet. By knowing when an opponent will raise his bet, our agent can obtain an estimate of the opponent's hand strength based on his action.

This is actualised by determining the opponent's hand strength at the end of each game, for each round that the opponent had raised on. The minimum hand strength that the opponent raised on, 60% for example, is determined to be the threshold. Hence, it can be deduced that a raise in any round of subsequent games implies with high probability that the opponent's hand strength for the round exceeds 60%. This offers a baseline to compare against in every round. Taking bluffs and unexpected behaviour into account, we decided to perform an averaging to prevent anomalies from affecting the threshold.

h_4 - Opponent Hand Strength Estimate

We estimate the opponent's hand strength based on the actions of the current opponent within the same game through an evaluation function using the raise multiplier (mRaise) and call multiplier (mCall) mentioned above in h_2. This heuristic is calculated as shown below:

$$h_4 = 50 * mRaise^X * mCall^Y \quad (4)$$

where X = Number of times opponent has raised and Y = Number of times opponent has called in this current round

At the beginning of each game, opponent hand strength is initialised to 50%. Whenever the opponent raises, h_4 is multiplied by mRaise. If the opponent calls, h_4 is multiplied by mCall instead. Variables X and Y are initialised each round, with h_4 carrying forward to the conclusion of the current game.

The round correlation of hand strength in the first three rounds (preflop, flop, turn) against the hand strength in the river round was calculated from 10,000 hands, and determined to be 0.316 (preflop), 0.666 (flop) and 0.796 (turn). We observe that the hand strengths are more evenly distributed as more community cards are revealed. Hence, we reason that actions in later rounds should have a higher emphasis, and should have a higher multiplier value.

Thus, the values of the two multipliers should be dependent on the current round. For example, mRaise is initialised to 1.35 and mCall to 0.9 for the logical player in the river round. The multipliers in the other rounds are calculated as:

$$mRaise = 1.0 + (1.05 - 1.0)RoundCorrelation \quad (5)$$

$$mCall = 1.0 + (1.0 - 0.4)RoundCorrelation \quad (6)$$

For example in the preflop round, the raise multiplier will be 1.0 + (1.25 - 1.0) * 0.316.

3.2 ϕ - Evaluation Function

We first prove that Bayes' theorem can be used to convert the hand strength of each player to the probability of our agent winning the game. Given that two players A and B have respective hand strengths HS_A and HS_B, the probability of A defeating B is given by:

$$P(Awins) = \frac{HS_A - HS_A HS_B}{HS_A + HS_B - 2HS_A HS_B} \quad (7)$$

Suppose that there are N players in this context, where N is a sufficiently large number of players that encompass the wide range of possible hole cards. The result of competition between each two different players have been definitely determined by nature. We can thus assume that n_A = $N*HS_A$ and n_B = $N*HS_B$ are the number of people defeated respectively by A and B. Letting E be the set of all N players, we can define two sets to represent the given information:

- $E_A = \{x \mid x \in E, A \text{ defeats } x\}$
- $E_B = \{x \mid x \in E, B \text{ defeats } x\}$

We know that $|E_A| = n_A$ and $|E_B| = n_B$, and that the entire event space can be represented by a subset of the product of E_A and E_B, since A and B can be element x respectively in E_B and E_A. Since any of the N^2 events are equally likely to happen, we can use the number of occurrences to calculate the probability of winning.

P(A wins) = P($B \in E_A$ & $A \notin E_B$)
= N_A (N - N_B) / (N_A(N-N_B) + N_B(N-N_A))
= HS_A (1 - HS_B) / (HS_A(1-HS_B) + (HS_B(1-HS_A))
= (HS_A - $HS_A HS_B$) / (HS_A + HS_B - 2$HS_A HS_B$)

4 Training Approach

We adopted a reinforcement learning approach for our agent to learn three thresholds: the fold-breakeven threshold, the fold threshold, and the raise threshold. These thresholds govern our agent's action decisions and are defined as follows:

The fold-breakeven threshold (FBT) is the win rate when the estimated loss of folding hand and continuing playing is approximately the same. At ϕ = FBT, our agent will fold at a probability equal to the round correlation. We first estimate the final pot if we play to the last round and, given the current pot, solve the following equation to calculate the FBT:

$$CurrentPot = [FinalPot*(1-FBT)]-[FinalPot*FBT] \quad (8)$$

The fold threshold (FT) is the threshold that we will immediately fold our hand. Thus, at $\phi \leq$ FT, our agent will fold with 100% probability. The raise threshold (RT) is the threshold at which we should declare a raise, hence our agent will raise with 100% probability if $\phi \geq$ RT. It is given that FT<FBT<RT. The probability of folding and raising scale down linearly to the round correlation as ϕ approaches FBT.

We train the fold thresholds by assigning an arbitrary number to the raise threshold and design two test agents with the fold threshold of 0.05 and 0.40 respectively. These two bots will play against each other for 500 games. At the end, the agent that has net gain is declared the winner, and the fold threshold of the loser will move closer to the fold threshold of the winner by 0.01. After the FT was found to converge at 0.23, this was fixed to calculate the raise threshold.

Utilising the same training method to obtain the optimal fold threshold, we assigned a raise threshold (RT) of 0.40 and 0.90 to the two test agents. Similarly, the two agents compete against one another and at every 500 games, the RT of the loser is moved closer to the winner by 0.01. The optimal value is found to be 0.6 after our rigorous testing.

5 Testing Results

We define a performance measure for our agent for comparison with three untrained agents (honest, passive and aggressive): Average Dollars Won per hand. This offers a quantification of how much the agent is able to win and how fast it was able to win. Due to the inability to determine the number of games played against each opponent, we are unable to determine the exact performance of our agent against other agents in the tournament, only whether it won or loss. The behaviours of the agents tested are as follows:

- **Honest Agent**: The following agents' evaluation function is solely based on one variable: its hand strength. The honest agent will raise above 60%, fold below 30%, and call on every hand strength between.
- **Passive Agent**: The passive agent is biased against raising, hence the threshold for raising and folding will be 70% and 40% respectively.
- **Aggressive Agent**: The aggressive agent is biased against folding, hence the threshold for raising and folding will be 50% and 10% respectively.

This performance metric is as shown below in Table 3.

Table 3: Poker Performance for Various Agents

Agent Battled	Average Dollars Won per Hand
Honest	4 (∼$2000 after 500 games)
Passive	5 (∼$2500 after 500 games)
Aggressive	2 (∼$1000 after 500 games)

From the results in the table above, it is clear that our agent performs best against the passive player as compared to the aggressive player. This is likely because of the former's high fold ratio, which makes it easy for our agent to exploit and increase its winnings.

6 Discussion

Given more resources, we would implement testing on human players to take advantage of psychological factors. Besides aggression, the range of starting hands that the opponent is willing play is another factor that we should take into account. Poker players are categorised into 4 categories based on aggression (passive/aggressive) and risk adversity (loose/tight), and have weaknesses that can be exploited with specific playstyles.

Given more time, we would also have explored the use of neural networks in the design of our agent. By making use of the unmatched predictive performance of advanced neural network models, it might have been possible for us to further train and improve on our agent. We would have tried to replicate the success of Liberatus given more resources [Brown and Sandholm, 2017]; the 25 million core hours required for training, as well as the sheer skill and understanding required, made it a far stretch for us to implement.

7 Conclusion

In this project, we have created an agent to play Two-Player Heads Up Limit Texas Hold'Em Poker. Through the use of novel heuristics and reinforcement learning, our agent estimates the probability of winning for both himself and the opponent while exploiting his opponent's behaviour to make sound betting decisions. While there exist points for improvement that could be solved by having more time, data, and resources, we believe that we have created a strong and innovative intelligent agent.

References

[Brown and Sandholm, 2017] Noam Brown and Tuomas Sandholm. Libratus: The superhuman ai for no-limit poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, Melbourne, Australia, August 2017.

[Routledge, 2005] Richard Routledge. Law of large numbers. 2005, https://www.britannica.com/science/law-of-large-numbers.

[Shackleford, 2017] Michael Shackleford. Two-player power ratings in texas hold 'em. 2017, https://wizardofodds.com/games/texas-hold-em/2-player-game/.

[TournamentTerminator.com, 2010] TournamentTerminator.com. Classification of poker playing styles / opponents. 2010, https://www.tournamentterminator.com/tournament-strategy/tournament-basics/typespoker-playing-styles-classifying-opponents/.

[West and Allen, 2018] Darrell M. West and John R. Allen. How artificial intelligence is transforming the world. Technical report, The Brookings Institution, 2018.