Developing An Intellegent Agent For Texas Hold'em Poker

Ali Berk Karaarslan
TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY, TURKEY
akaraarslan@etu.edu.tr

***Abstract*—This project aimed to develop a Poker game with an AI opponent implemented in Java. The Poker game features standard rules and gameplay mechanics, including card dealing, betting, and hand evaluation. The AI player utilizes algorithms based on probabilistic reasoning to make optimal decisions during gameplay. Overall, this project contributes to the intersection of game development and artificial intelligence, offering insights into the implementation of intelligent agents in gaming environments.**
***Keywords*—Poker, Monte Carlo, Betting, Cards.**

I. Introduction

Texas Hold'em Poker is a popular variation of the classic card game, typically enjoyed by 2 to 10 players. At the start of the game, each player is dealt two random cards from the deck, known as "hole cards." These cards are kept private and are not revealed to other players.

Following the initial deal, the game progresses through a series of betting rounds and community card reveals. Five community cards are dealt face-up on the table in three stages: the "flop" (three cards), the "turn" (one card), and the "river" (one card). Players use a combination of their hole cards and the community cards to form the best possible five-card hand.

The objective of Texas Hold'em Poker is to win chips, which represent money, from other players by having the strongest hand or by bluffing opponents into folding their hands. The player with the best hand, or the last player remaining after all others have folded, wins the pot—the total amount of chips wagered during the hand.

However, players must exercise caution, as losing all of their chips results in elimination from the game. Thus, strategic decision-making is paramount, as players must balance risk and reward to maximize their chances of success.

During each round of betting, players have several options at their disposal. They can "fold" their hand, forfeiting any chips already wagered and withdrawing from the current hand. Alternatively, they can "call" by matching the current bet, "raise" by increasing the bet amount, or "check" if no bet has been made.

We aim to develop a AI module, known as the PokerBot, capable of competing against human players and other AI opponents in Texas Hold'em Poker games. The PokerBot will utilize advanced algorithms and decision-making processes to analyze game dynamics, predict opponent behavior, and make strategic moves to gain an advantage.

II. Related Work

In the realm of artificial intelligence for poker playing, various methodologies have been employed to develop effective strategies. Understanding these methods can provide valuable insights into the diverse approaches utilized in similar projects. This section explores several prominent methods, including Counterfactual Regret Minimization (CFR), Table Method, Monte Carlo Simulations, Bayesian Network, and Deep Learning.

Counterfactual Regret Minimization (CFR):

CFR stands as a widely utilized technique in strategic games, aiming to compute the optimal strategy. It operates by assessing the decisions made by players during the game and computing a "regret" value based on the outcomes. Subsequently, it allows for improved decision-making in future rounds by leveraging these computed regret values. [1]

Table Method:

The Table Method offers a straightforward approach suitable for basic poker variations. It involves the creation of a lookup table encompassing all potential game scenarios. When faced with a decision, the AI refers to this precalculated table to make informed choices.

Monte Carlo Simulations:

Monte Carlo Simulations present a statistical method for predicting the probable outcomes of a system or event. By generating a set of random numbers based on a defined model, simulations enable the analysis of potential outcomes, facilitating informed decision-making by extrapolating likely results from simulated scenarios.

Bayesian Network:

Utilizing Bayes' theorem, Bayesian Network computes the probability of an event considering the information from related events. These relationships are represented in a graphical structure, with nodes representing variables and edges denoting dependencies between variables. [2]

Deep Learning:

Incorporating artificial neural networks, Deep Learning analyzes opponents' strategies and endeavors to make optimal decisions based on learned patterns. By employing models like Recurrent Neural Networks (RNNs), it predicts opponents' moves and employs techniques such as reinforcement learning to determine the best course of action.

Considering the methods discussed above, it is determined that Counterfactual Regret Minimization (CFR) and Deep Learning are not suitable for this project. The rationale behind this decision lies in the project's focus on creating an AI agent capable of making accurate decisions rather than one that learns over time. Additionally, constructing a comprehensive table for Texas Hold'em poker is practically unfeasible.

Therefore, the chosen approach for this project primarily revolves around Monte Carlo Simulations. This method aligns well with the project's objective of developing an AI agent capable of strategic decision-making in Texas Hold'em poker games.

III. Methodology

The development of the PokerBot AI module followed a systematic approach aimed at creating an effective and strategic player capable of competing in Texas Hold'em Poker game. This section outlines the methodology employed in the design, training, and evaluation of the PokerBot..

A. *Alghoritm Selection*

Monte Carlo simulations were integrated into the decision-making process of the PokerBot to enhance its strategic capabilities. This involved the following key steps:

1) Integration of Monte Carlo Simulations:

Monte Carlo simulations were selected as a fundamental component of the PokerBot's decision-making algorithm due to their ability to provide probabilistic insights into game scenarios.

By simulating numerous possible game outcomes, Monte Carlo simulations enabled the PokerBot to evaluate the expected value of different actions and make informed decisions based on strategic analysis.

2) Strategic Insights from Simulations:

The simulations allowed the PokerBot to explore a wide range of potential game states and anticipate the likely outcomes of different actions.

By assessing the expected value of each possible action, the PokerBot could prioritize moves that offered the highest probability of success, thereby enhancing its overall strategic performance.

3) Probabilistic Analysis:

Monte Carlo simulations facilitated probabilistic analysis of complex game situations, enabling the PokerBot to calculate the likelihood of success for each possible action.

This probabilistic approach enabled the PokerBot to adapt its strategy dynamically based on changing game conditions and opponent behaviors.

4) Optimal Decision-Making:

Through Monte Carlo simulations, the PokerBot could identify optimal decision-making strategies that maximized its chances of success in various game scenarios.

By selecting actions with the highest expected value, the PokerBot could navigate the complexities of Texas Hold'em Poker with greater precision and efficiency.

B. *Implementation Details*

B.1 Artificial Intelligence Working Principle:

As we mention earlier, the working principle primarily relies on Monte Carlo Simulation. As the AI can only perceive its own hand and the community cards, it conducts a certain number of simulations based on this information to estimate both its own and the opponent's potential card combinations. Subsequently, it interprets the results to make decisions.

It is mainly using this table to give score for each card combinations:

Card Combination	***Score***
Royal Flush	900 + HighCard
Straight Flush	800 + HighCard
Four Of A Kind	700 + FourOfAKind Rank + OtherCard
Full House	600 + ThreeOfAKind Rank + PairRank
Flush	500 + All Card Ranks
Straight	400 + HighCard
Three Of A Kind	300+ThreeOfAKind Rank +2 HighCard
Two Pair	200 + PairRanks + Other HighCard
Pair	100 + PairRank + Other HighCard Rank

Table 1. Card Combination Scores

B.2 Calculation of Estimated Card Combination Score:

In Texas Hold'em Poker, a total of seven cards are needed to accurately compute a player's hand combination. However, during gameplay, only two player cards and up to five community cards may be visible. That means in the middle of the game, player could not perceive all the seven cards. For that reason, to ensure a complete set of seven cards for calculation, additional cards are drawn from a new deck, totaling to seven cards (excluding duplicates of cards already in the player's hand).

With a new deck comprising seven cards, the score for each possible combination is calculated according to the scoring criteria previously mentioned.

The obtained score corresponds to a particular hand combination, which is then recorded in a HashMap by incrementing the count for the respective combination.

The previous three steps are iterated for a specified number of simulations (simulation count that we will mention later). Upon completion, the HashMap is populated with counts indicating how frequently each hand combination occurred.

Subsequently, the probabilities of each hand combination occurring are calculated as percentages based on the counts obtained.

With a compiled list of probabilities for each hand combination, the final step involves multiplying these probabilities by the corresponding hand combination scores and summing the products. This yields the "expected value" representing the anticipated score for the player's hand based on the initial input cards.

B.3 Estimation of Opponent Player's Card Combination Score:

Now that AI calculated its own expected card score, It will calculate the opponents expected scores. Since the AI cannot see the opponent's cards, it lacks sufficient data to accurately predict their hand combination. Therefore, an estimated card combination score is derived based on both the community cards and the opponent's actions.

Using the same method as "Calculation of Estimated Card Combination Score," an estimated score is computed based on the community cards.

The obtained estimated score is adjusted according to the opponent's actions. For this purpose, the potential actions of the opponent are considered:

Move	***Meaning***	***Amount***
Check	Indicates that, opponent's cards are not particularly strong.	+%20
Call	Indicates that, opponent's cards may be formidable	+%30
Raise	Indicates that, opponents cards may be strong depending on the raise amount	+%50/+%60
All-In	Indicates that, opponent's cards may be highly strong	+%65

Table 2. Score Increase Percentages Due To Opponent’s Move

The score obtained based on the opponent's action is then multiplied by these percentages to derive a new estimated score. The amount of "Raise" varies based on how much did the opponent bet its total money. If the opponent raises half of its money, then the increase amount will be +%55

Ultimately, the hand combination corresponding to the obtained score represents the estimated hand combination of the opponent.

B.4 Decision-Making Mechanism:

Now that AI could calculates its expected card combination score (Personal Score = PS), and could calculate the estimated card combination scores of all opponents (Opponent Score = OS) using the algorithms mentioned above. PokerBot makes decision with the relation between PS and OS.

The amount of difference between PS and OS indicates a substantial advantage for the player over their opponents in terms of hand strength or strategic position. This differences declared as intervals denoted as follows:

Intervals	***Difference***
Large Interval (>>>)	500
Middle Interval (>>)	150
Small Interval (>)	50

Table 3. Intervals

For instance, if the difference of two scores is over 500, it corresponds to “Large Interval (>>>)”. If the difference is 220, it corresponds to “Middle Interval (>>)”.

PokerBot determines its move based on this differences. The possible scenarios are as follows:

	Fold	***Check/Call***	***Raise***	***All-In***
PS >>> OS	%0	%0	%0	%100
PS >> OS	%0	%0	%100	%0
PS > OS	%0	%100	%0	%0
PS = OS	%9.3	%80	%10	%0.7
PS < OS	%0	%100	%0	%0
PS << OS	%50	%50	%0	%0
PS <<< OS	%100	%0	%0	%0

Table 4. Decision Possibilities Due To Personal And Opponent Scores

Decisions are made according to these scenarios. For instance, If PokerBot’s personal score is 700 and opponent’s expected score is 150, (This means PS >>> OS), PokerBot goes All-In.

The amount of "Raise" varies based on how much larger PS is compared to OS. If the gap between PS and OS is significantly large, then the raise amount is large too.

C. Graphical User Interface (GUI) Overview:

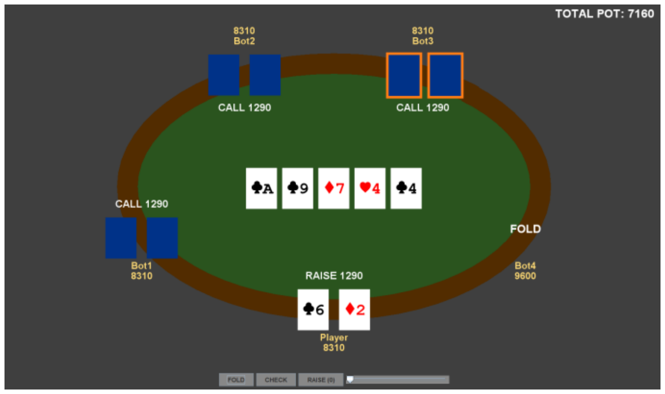

Fig.1 *Graphical User Interface(GUI) Of The Game*

At the heart of the interface is an elliptical representation of the poker table, the central arena where the game unfolds. Positioned around this table are the individual players, each with their own set of cards and corresponding budget. To maintain the secrecy of each player's hand, cards belonging to opponents remain concealed, appearing as dark blue rectangles.

Player Representation:

Each player is denoted by their name, their cards and current budget, providing essential information about the participants in the game.

Upon making a move, whether it be a bot or a human player, the action is displayed beneath or above the respective player's cards, depending on their position on the screen.

Action Buttons:

Located at the bottom of the interface are action buttons, providing players with various options to interact with the game. These buttons typically include actions such as fold, check, call, and raise. Additionally, a raise amount slider facilitates precise adjustments to the betting amount.

Community Cards:

As the game progresses, community cards are gradually revealed at the center of the table. These community cards contribute to the formation of hands and become increasingly important as the rounds unfold.

Game Progress Indicator:

Positioned on the top right corner of the screen is a display showing the total bet accumulated throughout the duration of the game, as well as the current stage of the game. Additionally, the indicator highlights the current player's turn with an orange frame around their cards.

Note: For a visual representation of the GUI elements, refer to Fig.1.

IV. PERFORMANCE

In this section, we present a comparative analysis of the performance of our PokerBot across varying Monte Carlo simulation counts.

The table below illustrates the results obtained by running our PokerBot with different Monte Carlo simulation counts. We will give two initial cards and we explore how increasing the number of simulations impacts the PokerBot's card estimation accuracy and execution time. By analyzing these results, we aim to identify the optimal simulation count that balances computational efficiency with strategic effectiveness.

Initial Cards	**(diamonds)2 (spades)4**	
Simulation Count	***Execution Time***	***Accuracy***
1,000	28 ms	***%96***
10,000	70 ms	**%97.5**
100,000	593 ms	**%98.2**
500,000	2609 ms	**%99**

Table.4 Simulation 1

Initial Cards	**(spades)7 (heart)5**	
Simulation Count	***Execution Time***	***Accuracy***
1,000	27 ms	***%95***
10,000	71 ms	**%96.6**
100,000	620 ms	**%97.3**
500,000	2680 ms	**%97.7**

Table.5 Simulation 2

After conducting extensive simulations with varying Monte Carlo simulation counts, including 1000, 10,000, 100,000, and 500,000, it became evident that the choice of simulation count significantly impacts both the accuracy and computational resources required by the PokerBot.

The simulations revealed that while higher simulation counts such as 100,000 and 500,000 resulted in marginally improved accuracy, they also incurred a substantial increase in computational overhead. The execution time for these higher counts was significantly longer, rendering them less practical for real-time decision-making in Texas Hold'em Poker games.

In contrast, the simulation with a count of 10,000 demonstrated a favorable balance between accuracy and computational efficiency. Despite offering slightly lower accuracy compared to the higher counts, the 10,000 simulations provided satisfactory performance while maintaining a reasonable execution time.

Considering the trade-off between accuracy and computational resources, the decision was made to adopt the Monte Carlo simulation count of 10,000 for the PokerBot. This choice ensures a good level of accuracy while keeping the computational overhead manageable, allowing the PokerBot to make informed and timely decisions during gameplay.

By selecting the 10,000 simulation count, we strike a practical balance between precision and efficiency, enabling the PokerBot to deliver optimal performance in Texas Hold'em Poker games

V. CONCLUSION

In conclusion, this project aimed to develop an effective and strategic Intellegent Agent named PokerBot capable of competing in Texas Hold'em Poker game. The methodology involved leveraging Monte Carlo Simulation to estimate optimal decision-making strategies based on limited information about the player's hand and the community cards. By simulating numerous game scenarios, the PokerBot evaluated the expected value of different actions, enabling it to select the most advantageous course of action through probabilistic analysis.

The main logic behind the decision-making process involved assessing the expected value of different actions based on the current state of the game and the PokerBot's hand. Using Monte Carlo Simulation, the PokerBot estimated the potential outcomes of various moves, considering factors such as the strength of its own hand, the community cards, and potential opponent strategies. By iteratively simulating gameplay scenarios and analyzing the resulting probabilities, the PokerBot made strategic decisions to maximize its expected value and optimize its chances of winning.

Overall, this project contributes to the field of AI gaming in developing strategic decision-making AI systems for complex games like Texas Hold'em Poker. The user-friendly GUI enhances the player experience, making the game accessible to both novice and experienced players alike.

References

[1] M. Zinkevich, M. Johanson, M. Bowling and C. Piccione, *Regret Minimization in Games with Incomplete Information*. *Jan 2008.*

[2] Stephenson and T. Andrew, An Introduction to Bayesian Network Theory and Usage, 2000.