Final Project:

AI poker agent

Date:	15/01/2021
Course:	Artificial Intelligence
Supervisor:	Yuantao Fan
Students:	M.Fahad Hassan
	Wahaj Afzal
	Sebastian Hagblom
Students IDs:	fahhas20
	wahafz20
	sebhag20

Abstract

The task of the project has been to design a memorizing poker agent where it utilizes the behaviour of the players at the same table. The features which were used for predictions were the strength of the opponents' hands at the showdown, as well as the corresponding bets. The input features are then used to predict the strength of the highest bidding opponent using a polynomial for exponential moving average analysis. The result of the implementation is an agent with the ability to adapt its strategy to the current players at the table. Unfortunately, the agent did not manage to win the extra times, and could have been possible by introducing bluffs and maximizing the utility of successful moves.

Index

1. Introduction

The game of poker that will be played, contains three to five players and is played from a server-client model. The initial state of each game is that every player is dealt 200 chips from the "dealer" and a hand consisting of five cards. The cards are labelled with ranks from 2-A and suits (heart, spades, diamonds and clubs) and could create different types which are presented in the figure below. The lower the probability of getting a certain combination, the higher are the chances of winning the current round. The game is over when there is only one player remaining and the other players are "bankrupt".

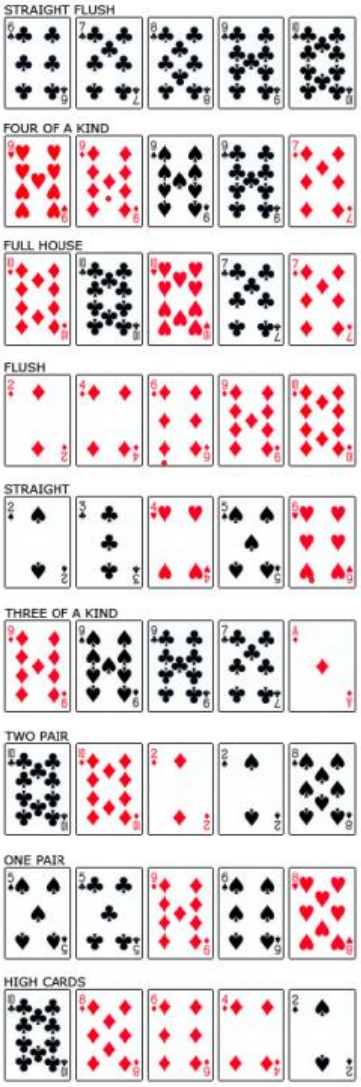

Figure 1: types of card ordered by most to less important [1].

1.1. Phases and Actions

The Poker game consists of seven different phases, where the first phase is the dealing phase. All the players are dealt five cards each to evaluate its strength and then make their actions or bets which is the second phase. In order to proceed to the next stage, all the players have to agree on the same amount of money put into the pot, if any player cannot fulfil these criteria the solution is then to fold and not be a part of the round. The actions that can be used to fulfil the criteria, are the following:

- **Check:** Does not bet any coins but are still participating, can only be made if no bid has been made.
- **Call:** Bidding the same amount as the current highest bid.
- **Raise:** Bids more than the current bet to still participate in the round.
- **All- in:** Agent does not have enough coins to bet or has very great cards, bets all the remaining coins.
- **Fold:** Agent throws in its cards and is out of the round.

When the first bidding round is done, the players get the opportunity to throw cards. The players can either throw all their cards and get new ones, do not throw any card or just only a few of them. After the exchange session, another bidding round is present and then proceeds to showdown where the player with the best cards wins the pot.

1.2. Agent types

The different methods that could be applied with artificial intelligence in order to play poker successfully are many, but the simplest type of agent is the random agent. The only thing the random agent is considering is the type of actions that can be made and thereafter randomizes the actions with no respect to their hand. One more advanced method than acting randomly is the reflex agent, where this type is considering their own strengths and acts according to any kind of input feature. Some examples of input features are the strength of its hand and how its opponents are behaving and tends to be more successful than the random agent.

Finally, the most sophisticated type of these agents is the memory agent which is storing and utilizing previous behaviour of its opponents and turns it to its advantage. Some examples of how the agent will be able to beat its opponents is by predicting strength of opponents' hands depending on their bets and predicting the best set of actions in order to maximize the profit of each round.

1.3. Related work

Card games are very common as online or mobile games. The poker game is one of the most studied games, since its popularity in the casinos. Cohen mentions a few strategies using AI agents to play poker and analyse its opponents, containing reinforcement learning, where Markov decision processes (MDPs) are used, and implementation of artificial neural networks to train the model of playing the game [2]. The same type of learning also resolved the uncertainties of poker (Texas' hold-em) using Partially Observable Markov Decision Processes (POMDPs) since the different outcomes of the game cannot be completely observable when the game depends on actions from more than one player [3].

Since the examples above are implementing artificial neural networks in order to train the poker agent, it will thereby not be a deeper explanation and implementation of these methods in this report, because it is out of topic for the course. Although, the general concepts and knowledge that there are different types of research for solving the problem will be taken into consideration when implementing the agent of this project.

2. Method

2.1. Strategy and methods

The base of the agent which will be playing in the tournament is a “reflex agent” and plays based on the strength of its own hand. The agent is able to decide what to bet, based on the strength but also decide what cards to exchange if needed. The more “advanced” strategy of this agent is to use information from the opponent’s previous behaviour in order to predict their strengths, this will be explained further in the Performance, Environment, Actuator and Sensor description in *2.2 Performance, Environment, Actuator & Sensor description (PEAS)*.

2.1.1. Reflex agent

The reflex agent (`reflex_bet()`) places a bet depending on the strength of its own hand. The amount of the money that will be raised corresponds to a percentage of the maximum bet that it can be done. If the agent’s coins are less than 50, it will only raise the minimum possible bet to stay in the game.

Depending on the strength of the agent's hand the raised bet is different, for small strengths the percentage raised is small and for stronger hands the raise is bigger. The percentage of a specific hand is randomly chosen to have an error of 2%. An example would be for the hand whose strength is 30, then the percentage to be raised is randomly chosen between 15% and 19% of the maximum bet at that point of the game.

2.2. Memory Agent

The memory agent chooses only one of the opponents to evaluate its hand depending on its bet. The opponent whose bet is the biggest, is selected to be evaluated and tries to calculate its strength. If the memory agent’s hand is bigger than the selected agent’s hand, the memory agent will be winning the game.

Although, the agent can change this strategy to evaluate every agent or choose the first agent who has entered the game. No external modifications are required on the following steps, since it has already been considered these options. If other agents do not have made a bet yet, the memory agent just takes the action to check.

When the opponent has been chosen; or in the case of all of them, one of them is selected; the opponent's name and bet is given to the prediction function for prediction of their hand.

The memory agent reacts as the following:

- If the opponent's predicted hand strength is 50% more than the memory agent's hand strength, the action taken is "fold".
- If a memory agent's hand strength is 50% more than the opponent's predicted hand strength, the action taken bet 30% of its coins.
- If the opponent's predicted hand strength is bigger than memory agent's hand strength and does not fulfil the first condition, it behaves as a reflex agent but betting 20% more than a normal reflex agent.
- In all other cases the look-up-tables strategy is used.

In order to generalize a description of what parameters are used and where it is applied, the Performance, Environment, Actuator & Sensor- description (PEAS) is presented in the table below and summarizes the sensors and actuators for execution in a specific environment.

Table 1: PEAS description of the agent

Performance	**Environment**	**Actuators**	**Sensors**
Prediction of opponents' hands and outplaying the strongest opponent at the table.	Poker table at the virtual poker application.	Actions which were mentioned in section *1.1 Phases and Actions* and the ability to exchange unwanted cards.	Given cards from their own hand, relative bets from opponents and the actual hand when proceeded to showdown.

2.3. Prediction function

The prediction function (`predict1()`) defines a low and high boundary of the bet, considering an error of 10% The values from previously saved data, which matches the agent name is received and evaluated if it is within the boundary of possible bets. Thirty such elements are chosen, which can be modified, to calculate the mean value of the hand of the specific opponent.

In case of not finding data of a specific agent for this bet, data of other agents of the same bet are chosen. If no values are found in the memory of the requested bet, the look-up-tables strategy is used. The mean value of the hand is given to the exponential moving average function to get with precision the hand strength, considering the opponents' strategy. Getting the value of the hand from the EMA function, it is compared to the memory agent's own hand, which is defined as the following.

- If the memory agent hand is much higher than the opponent's hand, (own hand strength is bigger than opponent's hand strength multiplied to 1.5), "TBR" massage is returned.
- If the memory agent hand is much less than the opponent's hand, (own hand strength multiplied to 1.5 is less than opponent's hand strength), "VMDS" massage is returned.
- If the opponent's hand strength is bigger than own hand, "eVOC" massage is returned.
- Otherwise, "VCC" massage is returned.

2.4. Exponential moving average value (EMA-value)

This function (`ex_moving_average()`) takes in consideration previously given average values of the same bet to calculate a value with precision. The exponential average value is calculated with the following formula:

$$\bar{x}_n = \frac{x_n + (1-\alpha)\cdot x_{n-1} + (1-\alpha)^2 \cdot x_{n-2} + \cdots}{1 + (1-\alpha) + (1-\alpha)^2 + \cdots} \quad (1)$$

Where $\underline{x}_n$ is the EMA value, α is chosen and $x_{n-k}, k \in Z$ are the previous average values.

Alpha at the beginning is 0.5, the k values goes from 0 to 11 and the formula has been adapted to the next one:

$$\bar{x}_n = \frac{num}{den} \quad (2)$$

$$num = \sum_{k=0}^{11} \left(1 - \alpha_{quotient\left(\frac{k}{3}\right)}\right)^k \cdot x_{n-k}$$

$$den = \sum_{k=0}^{11} \left(1 - \alpha_{quotient\left(\frac{k}{3}\right)}\right)^k$$

The value of α_0 is updated every time the showdown face has been reached and the cards have been shown. This is done to upgrade the values of the predictions and adjust to the real hand values. Every time an α is calculated the previous alpha is also saved but decreased in the position, with a maximum of four alphas, in this case. Therefore, at the beginning only α_0 exist with the default value of 0.5, and it gets adapted to the new results. x_{n-k} values also follow the same rule, at the beginning having only x_n, but each time the EMA function is called to calculate the mean value, the previous mean value is also saved (having a maximum number of 11).

The mean values depend on the bet and the alpha values are specific to each agent. The value of k can be modified but the number of alphas is relative to this k, having the same alpha for a group of three mean values.

When the alpha is need to upgrades, the latest three alphas are used as constant $(\alpha_1, \alpha_2, \alpha_3)$ and α_0 is calculated. This way the calculation complexity of the equation is also reduced, compared to having only one alpha for all 12 values. Furthermore, since opponents will not have a linear function of bet (having a specific hand, a specific bet is done which is invariable to time) the update will not totally adapt to the specific case and will not be so abrupt. This function will return the mean value of the opponent's hand.

2.5. Look-up-tables strategy

In this case the memory agent looks for the value of the specific opponent, its bet and memory agent hand's strength from the dictionary. This value will be the percentage of bet respect to a normal reflex agent. Default percentage is set to be at 60% if inputs got no value. This percentage is increased by 1% if winning is the result after taking an action, if its losses after taking an action percentage will decrease by 5%. When the value taken from look-up-tables is 0%, in that case action taken is "call". If the raised value is above 50% of the total agent's coins, the action taken is "call" instead of "bet".

2.6. Expected behaviour

The expected behaviour of the agent is to analyse and predict the strength of each player at the table, based on previous experience and then tries to beat the strongest player. After every round the prediction polynomial is updated depending on the eventual errors between estimation of hand strength and the actual result at showdown.

3. Results

3.1. Playing against reflex agents

The mirror images of the agent were created and 20 games were played among them. The final result after games is presented in the tables below. The reflex agents only evaluate their hands and execute actions and make bets within a specific range, depending on how great the cards are on hand.

Table 2: Final result of playing against memory/reflex agents for 20 games.

Agent name	Wins
FHSHWA01	9
FHSHWA02_reflex	5
FHSHWA03_reflex	6

3.2. Participation in the qualification tournament

In the qualifiers for the main tournament, the agent was sorted into the second group of five in total. The qualifiers consisted of 20 games where the agent with the most wins would proceed to the main tournament and the others were eliminated. The result from the qualifiers is presented in the table below as well as the performance of the agent specifically:

Table 3: Results from the qualifying tournament group 2

Agent Name	Wins
CASTILLEJOS_SANANIKONE	16
FHSHWA01	2
ALEX_JOHAN	2
Group_5	0

Table 4: Agent placements in the 20 games from the tournament

Placement	Frequency (20 games in total)
First	2
Second	11
Third	6
Fourth	1

4. Discussion

4.1. Observations from games against own agents

In order to evaluate the performance of the agent which will be entering the qualification tournament, it had to face two reflex agents. By the result in the second table, the memory agent manages to win the majority of the games which is considered as an acceptable performance. Since the reflex agents bid randomly within certain ranges, this also shows that the memory agent manages to predict and adapt their strategy at an acceptable level in order to succeed.

4.2. Observations from the qualification tournament and future improvements

By the result of the tournament, the performance of the agent was not as expected where the group thought of more wins than that it turned out. When analysing the results of each game the majority of results, the agent finished second place after the winners of group 7. Since the agent was fighting at the top during the majority of the games, the result is thereby looking better than what the final result is actually saying about the agent's performance in the tournament which is presented in the third and fourth table.

When analysing the result and the performance after the tournament, the agent could have been improved even further for better results. One example of an improvement is to try and pretend to have greater cards than the opponent and analyse their behaviour when they fold. The result of the strategy would be to win the smaller amounts which would be won undisputed since the opponents would eventually fold their cards.

Another aspect that should have been investigated further was the optimality of selecting certain in order to maximize the agent's utility and also reduce the utility of the opponents. The solution to this optimization problem would have been the expectiminimax algorithm where the opponents' actions are considered as chance nodes. The result of that type of implementation could cause the agent to make better and even fewer actions and increase its utility which is the number of coins won in each round.

5. Conclusions

In conclusion, the agent which has been designed to play a standard poker game from a server-client model performs great against own reflex agents when predicting strengths based on previous behaviour. On the other hand, when playing against more sophisticated agents, the agent tends to stay in the game until the final two but then does not manage to win as many games as expected. The improvements that could be made for the agent are better bluffing strategies, using search algorithms to find the maximum utility as an advantage and predicting losing paths for the opponents.

Bibliography

[1]. Halmstad University, "*AI Poker Project"*,

Available: https://bb.hh.se/bbcswebdav/pid-309964-dt-content-rid-2834024_1/courses/7710/AI_poker_project.pdf (Downloaded 4/01/2021)

[2]. Cohen, Ori (2019) "*Using Artificial Intelligence Methods To Win In Poker"*, Towards Data Science,

Available: https://towardsdatascience.com/using-artificial-intelligence-methods-to-win-in-poker-5559cc522f3b (Downloaded: 12/01/2021)

[3]. Oliehoek, Frans (2005) "*Game theory and AI: a unified approach to poker games*", University of Amsterdam,Available: https://www.google.com/url?q=http://people.csail.mit.edu/fao/docs/Oliehoek05MSc.pdf&sa=D&ust=1611075684772000&usg=AOvVaw3X5fuwAfPR4uWJhdXxmnDx (Downloaded: 11/01/2021)