Artificial Intelligence Project Report
DT8042 HT23, Halmstad University

Fabrice BODSON, Mery ENSAY, Morgane BOUVIER HAREL, Maciej JANKOWSI

January 2024

Table of content

1 Abstract

Type of the implemented agent
During this project, we had to develop AI agents, that would be able to compete in a Poker tournament and try to win. So, we implemented a reflex poker agent, which was able to check his hand, his money, to take the best action with these information. This agent was also able to sort its hand, and throw the cards that were not relevant for the current game state.

Underlying strategy
The stategy is focused on the strength of the hand the agent received from the server. Accordingly to a win probability associated with each possible hand, and the money it has, the agent is able to perform the best action to win the game hand. The agent doesn't take into account the action of the other players.

Our reflex agent will analyze its hand each time it has to play. If it has a hand with a high card, it is expected to discard this card when the server asks, as this card is the lowest one.

We used the PEAS description to define the components of our agent for this poker. The performance measure is the number of coins won per game, the environment is the game board (the other player's hands, coins, ...), the actuators are the actions taken depending on the hand and the sensors is the strength of the hand that has been identified. In section 3.2, we detailed our use of the PEAS description for our agent.

Performance during the tournament
Finally, during the pre-tournament, the reflex agent performed very well, as it won 15 games over 20. It appears that it had an aggressive way to bid, a behavior that finally forced the other agents to fold.

Of course, during the agent's development, we identified that it was often losing against the random agent. To increase its winnings, we made it more agressive and less passive by betting large amount of coins. This update of our agent helped our agent to win the pre-tournament even if this way was very risky.

2 Introduction

AI Poker is a strategic card game where players wager on the value of their hand, composed of five cards from a standard 52-card deck. The goal is to have the highest-ranking hand according to a predefined hierarchy, with rarer combinations being more valuable. In AI Poker, the players are replaced by autonomous computer programs, and the game follows the rules of five-card draw poker.

This is a game where players compete to win a pot of chips by making the best hand using five cards. The game is controlled by a server application that manages the game flow and communicates with players. Players connect to the server and receive their initial cards. This server is launched and started from a GUI. Each round of play consists of a forced bet, a betting round, a draw phase, a second betting round, and a showdown. During the betting rounds, players can make decisions to check (pass), open (make the first bet), fold (drop out of the hand), call (match the highest bet), raise (increase the bet), or go all-in (bet all their remaining chips). The showdown determines the winner based on hand rankings. The player with the best hand takes the pot. The game continues in rounds until only one player remains with chips. The remaining player is declared the winner.

In AI Poker rules, the game begins with an ante, a forced bet, followed by the distribution of five cards to each player. The betting rounds commence with the first player's choice between checking or opening. Subsequent players respond to the open by folding, calling, or raising. After the first betting round, a draw phase allows players to discard up to three cards and receive new ones, potentially improving their hand. The second betting round follows, with players again making informed decisions based on their hand strength and the actions of their opponents. If any players remain after the second betting round, they reveal their hands, and the player with the highest-ranking hand wins the pot. The chips in play represent the sum of all players' initial chip stacks. Six types of chips are used: 1-coin, 5-coin, 10-coin, 25-coin, 50-coin, and 100-coin.

Then, the settings for the hands are formed from five cards, and the player(s) with the highest-ranking hand wins the pot. Hands are ranked hierarchically, with the strongest hands being straight flush, four of a kind, full house, flush, straight, three of a kind, two pair, one pair, and high card. Suits do not impact hand ranking, except for determining flush and straight flush hands. When two players have identical hands except for suit, the pot is split evenly.

In this project, two pots can be created during a round: the main pot and one or more side pots. The main pot is typically used to determine the winner with the best hand, while side pots are created when a player goes all-in and has insufficient chips to cover the remaining bets. When a player goes all-in, their bet is placed into the side pot, and the remaining players must contribute an equal amount to the side pot (or all their remaining chips if they have folded). The main pot remains unchanged. At the showdown, the winner of the side pot is determined based on the highest-ranking hand among the players who contributed to the side pot. The winner of the main pot is determined based on the highest-ranking hand among the players who did not go all-in. Due to the existence of side pots, it is possible for multiple players to win in a single round, even if they have hands of different ranks.

In an AI Poker game, we can use various agent types and AI methods to achieve competitive performance. Reflex agents are a simple starting point, but they lack adaptability.

Is any relevant work being applied to this problem?

Several AI Poker agents have achieved remarkable results in tournaments against human and AI opponents. For instance, in 2017, a team of researchers from Carnegie Mellon University unveiled Libratus (Superhuman AI for Heads-up No-Limit Poker, 2017), a groundbreaking AI poker agent that achieved remarkable success in a high-stakes heads-up no-limit hold'em tournament against four top human professionals. Libratus's performance was unprecedented, consistently defeating its human opponents and demonstrating the potential of AI in mastering complex strategic games. Libratus employed a sophisticated hybrid approach that combined MCTS with deep reinforcement learning (DNNs). MCTS efficiently explored different betting strategies, while DNNs learned to evaluate the strength of hand combinations and predict opponent behaviors. This synergy allowed Libratus to adapt and improve its strategies over time, making it a formidable opponent.

Another example is Google's AlphaGo Zero (Yin et al., 2023). Developed by Google DeepMind, AlphaGo Zero first mastered the game of Go and subsequently applied its learning capabilities to poker, achieving impressive results in simulation studies.

3 Method

3.1 PEAS description

PEAS is an acronym used in artificial intelligence to describe the fundamental components of an intelligent system. Each letter in the acronym represents a key aspect of the system. Here's what each letter means:

	Describe	Application in project
(P) Performance measure	Performance measurement defines the criteria that determine how well the system performs in accomplishing its task.	Number of coins won per game
(E) Environment	The environment is the set of external elements with which the system interacts to accomplish its task.	Game board
(A) Actuators	They are responsible for implementing the decisions made by the intelligent system.	Choice of action (Fold, Bet, ...) depending on the hand
(S) Sensors	They collect data that will be used by the system to make decisions	Hand strength analysis

PEAS offers a conceptual structure for decomposing an intelligent system into its essential components, providing a clear understanding of the intelligent system's objectives, environment, actions and means of perception.

3.2 Strategy and methods employed

3.2.1 Reflex Agent

When the reflex agent receives an action request from the server, it first analyzes its hand and evaluates its strength.

With his sensors the agent will look at the type of hand he has: pair, double pair, flush, ... And thanks to his type of hand, his probability of winning.

```
WIN_PROBABILITY ={
    'High card' : 0.5012,
    'Pair': 0.4226,
    'Two pair' : 0.0475,
    'Three of a kind' : 0.021,
    'Straight' : 0.0039,
    'Flush' : 0.0020,
    'Full house' : 0.0014,
    'Four of a kind' : 0.00024,
    'Straight flush' : 0.00015,
}

def identify_hand(self):
    # Get the value for the non-value cards
    ranks_figure = {
        'T': 10,
        'J': 11,
        'Q': 12,
        'K': 13,
        'A': 14
    }

    # Get the ranks from the hand
    ranks = [card[0] for card in self.CurrentHand]

    # Straight Flush
    if utils.is_straight_flush(self.CurrentHand, ranks_figure):
        return "Straight flush"

    # Four of a kind
    if ranks.count(ranks[0]) == 4:
        return "Four of a kind"

    # Full House
    if utils.is_full_house(ranks):
        return "Full House"

    # Flush
    if utils.is_flush(self.CurrentHand):
        return "Flush"

    # Straight
    if utils.is_straight(ranks, ranks_figure):
        return "Straight"

    # Three of a kind
    if ranks.count(ranks[0]) == 3:
        return "Three of a kind"

    # 2 Pairs
    if utils.is_two_pairs(ranks):
        return "Two pair"

    # 1 Pair
    for i in range(len(ranks)):
        for j in range(i + 1, len(ranks)):
            if ranks[i][0] == ranks[j][0]:
                return "Pair"

    return "High card"

def analyse_hand_strength(self):
    """
    Analyzes a given hand based on its type and strength.
    Returns the strength of the hand.
    """

    hand_type = self.identify_hand()
    print(f"HAND TYPE {hand_type}")
    strength = WIN_PROBABILITY[hand_type]
    return strength
```

Another technique used is when the server asks to discard cards, our agent will chose to discard cards only if his sensors tells him he has only one high card.

```
def queryCardsToThrow(self):
print("Requested information about what cards to throw")
print(self.CurrentHand)

evaluation = self.identify_hand()
print(f"HAND TYPE {evaluation}")

# If it is a High Card, player throws the first cards of his hand
if evaluation == CARD_THROW:
    card = self.find_min()
    print(card)
    throw = card
    return throw

return ''
```

The agent choses to throw the smallest card in its hand.

3.3 Expected behavior

3.3.1 Reflex agent

Our reflex agent, having no memory, will play by analyzing its hand each time. If it has a hand with a high card, it is expected to discard a card when the server asks, and this card will be the lowest.

1. If player has a weaker or equal hand than High Card and cannot bet, he FOLD
2. If player has a better hand than High Card and weaker or equal to a Straight, he raises
3. If player has a better hand than High Card and weaker or equal to a Straight, he raises (alternative condition)
4. If player has a better hand or equal than Flush, he puts ALL IN
5. Otherwise he CALL

4 Experiment and tournament result

4.1 Observations and results

During the development of the reflex agent, we observed at first that it lost almost everytime against the random agent, even with a better hand than the random agent. So we looked into its behavior and noticed that it folded too soon and too often. In order to manage that we changed the condition of the reflex agent, so that it would be more agressive and less passive. After that, the agent didn't lose anymore against the random agent.

However, it was very difficult to predict how well it would perform against the agents of the other groups during the pre-tournament. We expected an average result, as our agent was a reflex one and we expected the other groups to have a better agent with learning and memory techniques.

So the result was beyond our expectations, as we won the pre-tournament.

4.2 Pre-tournament analysis

During the pre-tournament, our reflex agent won 15 games over 20, so it was a landslide victory. Our agent appears to be very brutal, with large bet in game. A risky way to play, but the adversary might often end up with not enough money to compete, so they have to fold. The correction previously mentioned about the fold behavior may had a significant role on our agent's performances.

After discussing with some other groups, they also seemed to used a reflex agent, so we can conclude that our reflex agent has the best behavior planification among the other groups, and that's why it won. Maybe, against a learning or memory agent, after a few hands of winning, our agent would lose, as the other agent would understand its strategy.

Finally, the strategy based on the probability associated with each possible hand was a good one, as the lower the probability was, the higher it was possible to win with, as the strong hands have a very low probability to occur. Moreover, it was also a way for the agent to know the probability for the other to have each hand.

4.3 Amelioration

To improve our agent and prevent it from making the same mistakes. We thought of adding a memory and a Q-learning system. The values used for learning would have been the outcome of a round and the hand strength where each round would have been a single learning example.

For example, when he has to bet, make him learn from his mistakes if he once bet too high and lost, make him bet for an identical hand less later.

5 Conclusion

During this project we had to develop an intelligent agent that was able to perform at a poker game tournament. The main objective was to do an agent, that was able to take the best decisions, with its knowledge.

In our case, we developed a reflex agent that was able to act accordingly to the current game state, with a predetermined action set. At first it wasn't able to learn or to have and use memory, but we focused on the aspects in the game that our agent should take into account to take decisions and to win.

Then we realized that the agent was folding too quickly when playing against a random agent so we made it more agressive by making higher bets. In the end, this strategy was good, as we won the pre-tournament, which proves that our agent was well defined.

As our reflex agent has good performances, we then tried to improve it, with Q-learning techniques and memory which allows the agent to know better which cards to throw and what bet to do. The values used for learning would have been the outcome of a round and the hand strength where each round would have been a single learning example. Unfortunately, we did not managed to finish it on time but it remains an interesting improvement to do.

6 References

[1] Q.-Y. Yin et al., ‘AI in Human-computer Gaming: Techniques, Challenges and Opportunities’, Mach. Intell. Res., vol. 20, no. 3, pp. 299–317, Jun. 2023, doi: 10.1007/s11633-022-1384-6.

[2] N. Brown and T. Sandholm, ‘Superhuman AI for heads-up no-limit poker: Libratus beats top professionals’, Science, vol. 359, no. 6374, pp. 418–424, Jan. 2018, doi: 10.1126/science.aao1733.