Knowledge-Based Systems Final Report

GTO Poker Bot: A Knowledge-Based Approach to Optimal Poker Strategy

Arne Noori, Ido Pesok & Wes Convery

1. Introduction

GTO Poker Bot solves the problem of finding an optimal poker strategy using knowledge-based approaches and fixed strategies. The bot simulates human-like gameplay by interpreting the game state visually and clicking buttons in the browser. It has two agent types: a fixed strategy agent based on game theory optimal principles and a Deep Q-Network (DQN) agent that learns through self-play.

We chose this project to explore how AI can solve the complex game of poker, which remains an unsolved challenge unlike chess and Go. Poker has elements of imperfect information, randomness, and complex game theory, making it a fertile area for AI research.

The bot is implemented in Python and can work on various poker platforms. It uses the GPT-V model to interpret game state from screenshots and has a modular design with the fixed strategy agent in fixed.py and DQN agent in dqn_agent.py. Key parameters include a 52-card deck, predetermined starting stacks, and small/big blind amounts.

After running 1000 simulated hands, the fixed strategy agent won 927 hands, the DQN agent won 51, the random agent won 6, with 16 ties. This shows the fixed strategy agent, based on game theory optimal principles, significantly outperforms the learning-based DQN agent and random play. However, the DQN agent's poor performance suggests limitations in its ability to learn an effective strategy in this complex domain.

2. Problem Specification

The GTO Poker Bot project addresses the challenge of creating an AI agent that can play poker optimally by visually interpreting the game state and taking actions through a web browser, simulating human play. Poker presents unique difficulties for AI due to its elements of imperfect information, stochastic events, and the need for complex strategic reasoning.

Unlike perfect information games such as chess and Go, poker involves hidden information (opponents' private cards), which makes it challenging to determine the optimal strategy. The game also includes random events in the form of card dealing, requiring the AI to reason under uncertainty. Furthermore, poker demands sophisticated strategic thinking, including the ability to bluff, model opponents, and adapt to changing game dynamics.

The main challenges in developing a poker AI include:

1. Handling imperfect information: The AI must make decisions based on incomplete knowledge of the game state, considering the range of possible opponent holdings and strategies.
2. Dealing with stochastic events: The random dealing of cards introduces uncertainty, requiring the AI to reason probabilistically and adapt its strategy accordingly.
3. Modeling opponents: To make optimal decisions, the AI must be capable of inferring opponents' likely holdings and strategies based on their actions.
4. Developing a robust and adaptive strategy: The AI needs to employ a strategy that is effective against a wide range of opponent playing styles and able to adjust as the game progresses.

5. Interacting with the environment: The AI must be able to interpret the game state visually and execute actions through a web browser interface, simulating human interaction with the poker platform.

While significant progress has been made in poker AI, with notable achievements like Libratus [1], Pluribus [2], and DeepStack [4], poker remains an unsolved domain compared to games like chess and Go. Existing approaches often rely on computationally intensive algorithms and require direct access to the game state, making them unsuitable for deployment in real-world online poker environments.

The GTO Poker Bot project aims to tackle these challenges by developing an AI agent that can learn and execute optimal strategies in a human-like manner, interacting with the poker platform through visual interpretation and simulated actions. The project explores the potential of combining knowledge-based approaches, such as a fixed strategy based on game-theoretic principles, with reinforcement learning techniques to create an effective and adaptable poker AI.

Tools and Starting Points

The project leverages TensorFlow for implementing the DQN agent and GPT-V for interpreting the game state from screenshots. The fixed strategy agent [fixed.py] and the DQN agent [dqn_agent.py] serve as the core components, with the operational framework [play.py]) enabling interaction with online poker games.

PEAS

- **Performance:** Success is measured by the agent's ability to win hands and maximize earnings over time.
- **Environment:** Online poker platforms, characterized by varying game states, opponent strategies, and the visual presentation of information.
- **Actuators:** Actions within the game, such as fold, call, and raise, executed through simulated mouse clicks and keyboard inputs.
- **Sensors:** Screenshots of the game, processed and interpreted to understand the current state of the board and make decisions.

Game Rules and Parameters

Poker, specifically the Texas Hold'em variant, involves players making the best hand from two private cards and five community cards. The game consists of several betting rounds (preflop, flop, turn, river), with players having the option to fold, call, or raise. The complexity of poker comes from the strategic decisions made with incomplete information about opponents' hands.

Implementation-Specific Parameters

- **Number of Cards:** 52-card deck.
- **Starting Stack:** Each player begins with a predetermined amount of chips.
- **Blinds:** Small and big blinds are posted at the beginning of each hand to initiate betting.

3. Related Work

Several notable poker AI systems have been developed in recent years, each employing different approaches to tackle the complexities of the game. Libratus, developed at Carnegie Mellon University, uses a form of counterfactual regret minimization (CFR+) to compute strategies in real-time [1]. Pluribus, created by researchers from Facebook AI and Carnegie Mellon, builds upon the success of Libratus to beat elite human pros in multiplayer no-limit Texas hold'em [2]. DeepStack, developed by researchers from the University of Alberta and Czech Technical University, uses deep learning and decomposition to reason about game situations [13].

While these state-of-the-art systems have achieved remarkable success, they rely on sophisticated techniques that may be challenging to implement and computationally expensive. In contrast, the GTO Poker Bot explores the potential of a knowledge-based approach, particularly through the use of a fixed strategy agent.

This agent embodies the principles of game theory optimal play, providing a strong baseline for evaluating the performance of the learning agent (DQN).

The fixed strategy agent draws inspiration from the concept of Nash equilibrium, a central concept in game theory. In a two-player zero-sum game like heads-up poker, a Nash equilibrium is a pair of strategies where neither player can improve their expected outcome by unilaterally changing their strategy [14]. The fixed strategy agent aims to approximate a Nash equilibrium strategy by making decisions based on hand strength, pot odds, and opponent actions.

The implementation of the fixed strategy agent builds upon existing knowledge in the field of poker game theory. Resources such as GTO Wizard [15] and GTOBase [16] provide insights into game theory optimal strategies for various poker situations. By incorporating this knowledge into the fixed strategy agent, the GTO Poker Bot explores the effectiveness of a knowledge-based approach in achieving strong poker performance.

4. Implementation

The GTO Poker Bot is implemented using a modular architecture, with the main components being the Fixed Strategy Agent (`fixed.py`), the DQN Agent (`dqn_agent.py`), and the Operational Framework (`play.py`).

Fixed Strategy Agent

The Fixed Strategy Agent embodies the knowledge-based approach in the GTO Poker Bot. It makes decisions based on a predefined set of rules derived from game-theoretic optimal (GTO) principles. The agent considers factors such as hand strength, pot odds, and opponent actions to determine the optimal action in a given situation.

The implementation of the Fixed Strategy Agent involves:

1. Hand Strength Evaluation: The agent assesses the strength of its hand using a heuristic scoring system that assigns higher values to stronger hands (e.g., royal flush, straight flush, four of a kind) and lower values to weaker hands (e.g., high card, one pair). The evaluation takes into account both the private cards and the community cards.

2. Pot Odds Calculation: The agent computes the pot odds, which represent the ratio of the current size of the pot to the cost of a contemplated call. This information is used to determine whether it is profitable to continue in the hand or fold.

3. Opponent Action Consideration: The agent takes into account the actions of the opponents, such as their betting patterns and the frequency of their raises. This information is used to adjust the agent's strategy and make more informed decisions.

4. Decision Making: Based on the hand strength, pot odds, and opponent actions, the agent selects the optimal action from a predefined set of options (e.g., fold, call, raise). The decision-making process is guided by a set of rules that aim to maximize the expected value (EV) of the agent's actions.

The Fixed Strategy Agent's implementation is contained in the `fixed.py` file, which includes functions for hand evaluation, pot odds calculation, and decision making based on predefined rules.

DQN Agent

The DQN (Deep Q-Network) Agent is a reinforcement learning-based approach that learns to make optimal decisions through trial and error. The agent uses a neural network to approximate the action-value function, which estimates the expected cumulative reward for taking a particular action in a given state.

The implementation of the DQN Agent involves:

1. State Representation: The game state is represented as a vector that includes information about the agent's private cards, the community cards, the pot size, and the actions taken by the opponents. This state representation serves as the input to the neural network.

2. Action Space: The agent's action space consists of three possible actions: fold, call, and raise. These actions are represented as discrete values (e.g., 0, 1, 2) and are the outputs of the neural network.
3. Reward Function: The reward function determines the immediate reward the agent receives for taking a particular action. In the context of poker, rewards can be based on the amount of money won or lost in a hand.
4. Experience Replay: To stabilize the learning process, the agent uses an experience replay buffer that stores past state-action-reward-next_state tuples. During training, the agent samples mini-batches from this buffer to update the neural network parameters.
5. Training Process: The DQN Agent is trained through self-play, where it plays against itself or other agents to generate training data. At each step, the agent selects an action based on the current state, observes the reward and the next state, and stores this experience in the replay buffer. Periodically, the agent samples a mini-batch from the buffer and updates the neural network parameters using the Q-learning algorithm.

The DQN Agent's implementation is contained in the `dqn_agent.py` file, which includes the definition of the neural network architecture, the experience replay buffer, and the training loop.

Operational Framework

The Operational Framework (`play.py`) integrates the Fixed Strategy Agent and the DQN Agent with the poker environment. It handles the interaction between the agents and the poker platform, including game state interpretation and action execution.

The main components of the Operational Framework are:

1. Game State Interpretation: The framework uses the GPT-V model to interpret the game state from screenshots of the poker platform. It processes the visual information to extract relevant features such as the community cards, the agent's private cards, the pot size, and the opponents' actions.
2. Action Execution: Based on the decisions made by the agents, the framework simulates the appropriate actions (fold, call, raise) by sending commands to the poker platform through web browser automation techniques (e.g., Selenium, Puppeteer).
3. Agent Integration: The framework allows for seamless integration of different agent implementations, such as the Fixed Strategy Agent and the DQN Agent. It provides a unified interface for the agents to receive game state information and return their chosen actions.

The Operational Framework's implementation is contained in the `play.py` file, which includes functions for game state interpretation, action execution, and agent integration.

To evaluate the performance of the GTO Poker Bot, the framework includes evaluation scripts that simulate games against different types of opponents (e.g., random agents, fixed strategy agents) and measure the agents' performance in terms of win rate, total chips won, and other relevant metrics. These evaluation scripts are contained in files such as `evaluate.py` and `simulate.py`.

The GTO Poker Bot's implementation leverages various open-source libraries and tools, such as TensorFlow for building and training the DQN Agent, OpenCV for image processing, and Selenium or Puppeteer for web browser automation. The project's modular architecture allows for easy experimentation with different agent implementations and hyperparameters, facilitating the exploration of various approaches to developing an effective poker AI.

5. Analysis

Our goal was to create a poker bot that could consistently outperform a random agent and achieve a positive win rate using a knowledge-based approach. Specifically, we aimed for the fixed strategy agent to win at least 80% of hands against the random agent.

To evaluate the system, we conducted the following analyses:

1. Head-to-head comparison of the fixed strategy agent vs DQN agent over 1000 simulated hands. Performance metrics include number of hands won, total chips won, and overall win rate.
2. Performance of fixed strategy agent against human players of varying skill levels, measuring its profitability and adaptability.
3. Comparison of the fixed strategy agent's decisions to game theory optimal strategies from GTO Wizard [5] and GTOBase [6] to assess how closely it approximates optimal play.
4. Testing the fixed strategy agent's robustness and adaptability against a range of game situations and opponent strategies.

The results showed the fixed strategy agent performed strongly against both the DQN and human players. Its knowledge-based approach, considering hand strength, pot odds, and opponent actions, proved effective in making profitable decisions across different scenarios.

However, the fixed strategy's reliance on predefined rules and heuristics limited its performance in certain complex situations. This highlights the need for more advanced techniques like real-time adaptation and opponent modeling.

The DQN agent's poor performance, winning only 51 hands compared to the fixed strategy's 927 (Table 1), was a surprising result. This suggests the learning-based approach struggled to converge on an effective strategy in this complex domain, at least with the amount of training data and model architecture used. Investigating alternative deep learning techniques and gathering more diverse training data could potentially improve the DQN's performance.

Table 1: Number of hands won by each agent over 1000 simulated hands

Agent	Hands Won
Fixed Strategy	927
DQN	51
Random	6
Ties	16

The system ran efficiently, with 1000 hands simulated in a reasonable time. A key finding is the strong performance of the knowledge-based fixed strategy compared to the learning-based DQN and random agents. However, the fixed strategy's limitations suggest a combined approach, integrating knowledge-based models with learning-based techniques, could be a promising direction for future work.

6. Ethical Considerations

1. Fairness and integrity: The use of autonomous bots disrupts the level playing field expected by human players. Poker is based on human skill, psychology, and unpredictability, so introducing an AI that can play more effectively than humans undermines the spirit of fair competition.

2. Violation of terms of service: Most online poker platforms prohibit the use of bots or automated systems. Using such technology breaches these terms and can result in penalties like account suspension or legal action.

3. Transparency and disclosure: If deployed in real poker environments, it is important to disclose the presence of the GTO Poker Bot to other players. Transparency about the use of AI promotes trust and allows players to make informed decisions about their participation.

4. Security and privacy: The bot's ability to bypass cheat detection mechanisms challenges the security measures of online platforms. This capability could encourage the development of similar exploitative technologies. Additionally, the bot's operation may involve analyzing potentially sensitive game data and player behavior, raising privacy concerns.

5. Societal impact: The existence of poker bots could contribute to a broader normalization of cheating and unethical behavior in online environments, eroding trust in these platforms. Moreover, the use of such bots could disrupt the economy of online poker, which serves as a source of income for professional players and revenue for the platforms.

To mitigate these ethical risks, several strategies can be employed: - Transparency and consent in research environments, ensuring all participants are aware of and agree to the use of AI agents - Compliance with legal and ethical standards set by platforms and the broader community - Responsible disclosure of any discovered vulnerabilities to the affected parties - Engagement in public discourse to shape guidelines and regulations around the use of AI in online gaming

In conclusion, while the GTO Poker Bot represents a significant technological achievement, it underscores the need for a careful, ethical approach to the development and deployment of AI in real-world systems. As AI continues to advance, it is crucial that ethical frameworks and policies evolve to ensure these technologies are benefiting society while upholding principles of fairness, privacy, and security.

7. Link to code

https://github.com/arnenoori/gto-poker-bot

Tutorial on how to run the code and replicate our results are within the README.md file on the repository.

8. Summary

The GTO Poker Bot project aimed to create an AI agent that could play poker optimally by simulating human-like interactions with a web-based platform. The system consisted of a Fixed Strategy Agent based on game-theoretic principles and a Deep Q-Network (DQN) Agent that learned through self-play. The project utilized Python, TensorFlow, web automation tools, and the GPT-V model for game state interpretation.

Evaluation through simulations revealed that the Fixed Strategy Agent significantly outperformed the DQN Agent and a random agent, winning 927 out of 1000 hands. The DQN Agent's poor performance, winning only 51 hands, highlighted the challenges of learning-based approaches in this complex domain.

Compared to the Project Update submission, the current system features improved implementation and evaluation, including the integration of the GPT-V model and a modular architecture. The extensive simulations provided valuable insights into the strengths and limitations of different approaches.

Future improvements could include opponent modeling, advanced reinforcement learning algorithms, real-time adaptation, and expanded training data for the DQN Agent. Ethical considerations surrounding fairness, transparency, and responsible use in online poker environments should also be addressed.

In conclusion, the GTO Poker Bot project demonstrates the potential of knowledge-based approaches in poker AI while highlighting the challenges of learning-based methods. Although the Fixed Strategy Agent achieved impressive results, further research is needed to create truly robust and adaptive poker AI systems. By addressing the identified limitations and ethical considerations, this project serves as a foundation for future work in this exciting field.

9. References

[1] Brown, Noam, and Tuomas Sandholm. "Superhuman AI for Heads-up No-Limit Poker: Libratus Beats Top Professionals." Science, 17 Dec. 2017.

[2] "Pluribus: The First AI to Beat Pros in 6-Player Poker." Meta AI, 11 July 2019, ai.meta.com/blog/pluribus-first-ai-to-beat-pros-in-6-player-poker.

[3] Moravčík, Matej, et al. "DeepStack: Expert-Level Artificial Intelligence in Heads-up No-Limit Poker." Science, vol. 356, no. 6337, 2017, pp. 508-513.

[4] Nash, J. F. "Equilibrium Points in N-Person Games." Proceedings of the National Academy of Sciences, vol. 36, no. 1, 1950, pp. 48-49.

[5] "GTO Wizard: GTO Trainer for Cash Games & Tournaments." GTO Wizard, www.gtowizard.com.

[6] "GTO Poker Strategy Viewer, Trainer and HH Analyzer." GTOBase, 20 Jan. 2023.