CSC 487 Deep Learning Final Report

GTO Poker Bot - A Deep Reinforcement Learning Approach to Optimal Poker Strategy

Arne Noori, Ido Pesok, and Wes Convery

1. Project Description:

This report presents GTO Poker Bot, an AI system designed to find an optimal poker strategy using deep reinforcement learning. The bot operates at the browser level, simulating human-like gameplay by interpreting the game state visually. It offers 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. The project aims to explore the potential for AI to solve the complex game of poker, which, unlike chess and Go, remains an unsolved challenge. The bot is implemented in Python using the TensorFlow library and can be easily adapted to work on various poker platforms.

Poker presents a unique challenge for artificial intelligence due to its elements of imperfect information, stochasticity, and game-theoretic complexity. While AI has achieved superhuman performance in perfect information games like chess (BigFish) and Go (AlphaZero), poker remains an unsolved problem with room for further strategic optimization. This project, GTO Poker Bot, aims to tackle this challenge by developing an AI system that can learn and execute optimal poker strategies through deep reinforcement learning.

The primary motivation behind this project is to explore the potential for AI to master the intricate decision-making processes required in poker. By creating a bot that operates at the browser level and interprets the game state visually, we simulate human-like gameplay, enabling the AI to interact with various poker platforms without relying on direct access to the game's internal state. This approach opens up possibilities for testing and refining poker AI in real-world settings.

The GTO Poker Bot offers two distinct agent types: a fixed strategy agent and a DQN agent. The fixed strategy agent bases its decisions on game theory optimal principles, executing a predefined strategy that takes into account hand strength, pot odds, and opponent actions. On the other hand, the DQN agent learns its strategy through self-play, adjusting its decision-making based on the rewards it receives. By comparing the performance of these two agents, we can evaluate the effectiveness of reinforcement learning in approximating optimal poker strategy.

The implementation of GTO Poker Bot utilizes Python and the TensorFlow library. The bot is trained on a dataset of 1000 poker hands, enabling it to learn and adapt its strategy over time. Users can run the "play.py" script to observe the bot playing poker autonomously, making decisions based on the visual interpretation of the game state.

2. Project Design:

1. Embodied Agent (gamer folder): This module enables the bot to interact with the poker game at the browser level. It captures the game state through visual screenshots, which are then processed by GPT-V to extract relevant information such as community cards, hole cards, pot value, and player actions. This allows the bot to operate independently of the specific poker platform, making it adaptable to various online poker rooms.
2. Fixed Strategy Agent (fixed.py): This agent implements a fixed strategy based on game theory optimal principles. It considers factors such as hand strength, pot odds, and opponent actions to make decisions. The fixed strategy serves as a baseline for evaluating the performance of the learning agent.

3. DQN Agent (dqn_agent.py): The Deep Q-Network agent utilizes reinforcement learning to learn an optimal strategy through self-play. It employs a neural network to approximate the Q-value function, which estimates the expected long-term reward for each action in a given state. The agent updates its Q-network based on the rewards it receives during gameplay, allowing it to improve its decision-making over time.
4. Evaluation (evaluate.py): This module facilitates the comparison of the DQN agent's performance against the fixed strategy agent. It simulates a specified number of poker hands and tracks the wins, losses, and ties for each agent. The results are visualized through plots, providing insights into the relative strengths of the two approaches.

3. Implementation:

The GTO Poker Bot is implemented using Python and leverages several libraries and frameworks:

- TensorFlow: Used for building and training the DQN agent's neural network.
- OpenCV: Employed for image processing and analysis of the game state screenshots.
- PyAutoGUI: Utilized for simulating user interactions with the poker platform, such as clicking buttons and placing bets.
- NumPy: Used for efficient numerical computations and data manipulation.
- Matplotlib: Employed for visualizing the evaluation results and performance metrics.

The main flow of the system is as follows:

1. The embodied agent captures a screenshot of the poker game using the gamer module.
2. The screenshot is processed by GPT-V to extract relevant game state information.
3. The extracted game state is passed to the selected agent (fixed strategy or DQN) for decision-making.
4. The agent determines the optimal action based on its strategy and the current game state.
5. The embodied agent executes the chosen action by simulating user interactions with the poker platform.
6. The process repeats until the specified number of hands or a termination condition is reached.

The user can interact with the GTO Poker Bot through a command-line interface. The "play.py" script allows the user to specify the desired agent (fixed strategy or DQN), the number of hands to play, and other configuration options. The bot then autonomously plays poker, making decisions based on the visual interpretation of the game state.

4. Testing and Evaluation:

The GTO Poker Bot undergoes rigorous testing to ensure its reliability and performance. The evaluation module (evaluate.py) plays a significant role in this process. It simulates a large number of poker hands (e.g., 1000) between the DQN agent and the fixed strategy agent, keeping track of the wins, losses, and ties for each agent.

The evaluation results are visualized using Matplotlib, generating plots that showcase the cumulative wins and ties over the course of the simulated hands. These plots provide valuable insights into the relative performance of the two agents and help identify any trends or patterns in their decision-making.

In addition to the head-to-head comparison, the GTO Poker Bot is also tested against a random agent (random_agent.py). This helps assess the effectiveness of both the fixed strategy and the DQN agent in exploiting a purely random opponent.

5. Analysis:

The development of the GTO Poker Bot presented several challenges and opportunities for growth. One of the primary difficulties was creating an embodied agent that could accurately interpret the game state from visual information alone. This required extensive research and experimentation with image processing techniques and computer vision libraries. The use of GPT-V for extracting game state information proved to be a crucial component in overcoming this challenge.

Another significant hurdle was designing and training the DQN agent to learn an optimal strategy through self-play. Reinforcement learning algorithms can be notoriously unstable and sensitive to hyperparameter tuning. Extensive experimentation and iterative refinements were necessary to achieve a stable and effective learning process.

The fixed strategy agent, while serving as a valuable baseline, also presented its own set of challenges. Developing a robust strategy that could handle a wide range of game situations required a deep understanding of game theory optimal principles and poker-specific heuristics. Striking the right balance between exploitation and exploration was crucial to create a competitive fixed strategy.

Despite these challenges, the GTO Poker Bot demonstrates the potential for AI to tackle complex problems in imperfect information games like poker. The combination of the embodied agent, fixed strategy, and DQN learning approaches provides a comprehensive framework for exploring and refining optimal poker strategies.

Future extensions and improvements:

- Incorporating more advanced game state features and player modeling techniques to enhance decision-making.
- Exploring alternative reinforcement learning algorithms, such as Actor-Critic methods or Monte Carlo Tree Search, to improve learning efficiency and performance.
- Expanding the bot's capabilities to handle multi-player scenarios and different poker variants.
- Integrating online learning mechanisms to allow the bot to continuously adapt and improve its strategy based on real-time gameplay data.
- Enhancing the user interface and visualization tools to provide more intuitive insights into the bot's decision-making process.

In conclusion, the GTO Poker Bot project demonstrates the application of deep reinforcement learning and embodied AI in the pursuit of optimal poker strategies. By combining visual interpretation, fixed strategy heuristics, and self-play learning, the bot offers a promising approach to tackling the complexities of imperfect information games. While challenges remain, the project lays the foundation for further research and development in the field of poker AI, with potential implications for other domains involving strategic decision-making under uncertainty.

Code Repository:

The complete codebase for the GTO Poker Bot project can be found at: https://github.com/arnenoori/gto-poker-bot

The repository includes detailed instructions on how to set up and run the bot, as well as guidelines for replicating the results presented in this report.