Reinforcement Learning for Poker

Gabriel Larson

Abstract

This paper presents a reinforcement learning approach to playing poker, using the Keras-RL [1] library's built-in agents and the clubs_gym Python package [2] for simulating the poker environment. The project aims to explore the effectiveness of various reinforcement learning algorithms for mastering the complex game of poker. Through a series of experiments, we evaluate the performance of different agents and propose potential improvements to further enhance their capabilities.

1 Introduction

Poker is a popular card game with numerous variations, each presenting unique challenges and requiring different levels of skill and strategy. No Limit Texas Hold'em, a popular variant of poker, is a game of imperfect information, where players must make decisions based on incomplete knowledge of their opponents' cards and actions. This inherent complexity makes poker an intriguing problem for artificial intelligence (AI) research. The game's unique structure and the strategic depth required to excel at it have led to significant interest in the development of AI agents capable of mastering poker.

In this project, we focus on 9-player No Limit Hold'em, a complex variation where each player receives two private hole cards and shares five community cards with other players. The objective is to make the best five-card hand using a combination of hole and community cards. Players can bet any amount of chips at any time during their turn, with no upper limit, making the game highly dynamic and requiring adaptive strategies.

Reinforcement learning (RL) is a branch of machine learning that focuses on training agents to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. In recent years, reinforcement learning has emerged as a promising approach

to tackle complex problems, including those involving imperfect information, such as poker. By learning from trial and error, RL agents can adapt their strategies over time, eventually converging on optimal or near-optimal solutions.

The objective of this project is to explore the application of reinforcement learning techniques to the game of 9-player No Limit Hold'em, using the Keras-RL library and the clubs_gym Python package. Keras-RL provides a range of pre-built RL agents that can be easily integrated into various environments, while clubs_gym offers a flexible and extensible platform for simulating poker games. By combining these tools, we aim to study the effectiveness of different RL algorithms in learning poker strategies and adapting to diverse game situations.

The contributions of this paper include:

- An overview of related work in the field of AI and poker, discussing existing solutions and their limitations.
- A concise description of our approach, detailing the reinforcement learning algorithms and techniques employed.
- A thorough evaluation of our proposed solution through a series of experiments, analyzing the performance of different agents and their ability to learn effective poker strategies.
- Suggestions for future work to further study the performance of reinforcement learning agents in poker.

This paper is organized as follows: In Section 2, we provide a brief review of related work in the field of AI and poker, discussing existing solutions and their limitations. Section 3 describes our approach to the problem, including details of the reinforcement learning algorithms and techniques used. In Section 4, we present our experimental design and the results obtained from our agents. Section 5 provides an analysis of the results, followed by a conclusion and suggestions for future work in Section 6. Finally, we provide a bibliography for further reference.

2 Literature review

2.1 AI and Poker

AI research in poker has been an active area of study, given the game's complexity and its nature as an imperfect information game. One of the most significant milestones in this domain is the development of Libratus by Brown and Sandholm [3], which defeated top human professionals in heads-up no-limit Texas Hold'em poker. Libratus employed a combination

of abstraction techniques, endgame solving, and nested subgame solving to achieve its remarkable performance.

Another notable work is DeepStack by Moravčík et al. [4], which introduced an algorithm that combines recursive reasoning with deep learning to approximate the Nash equilibrium strategy in heads-up no-limit Texas Hold'em. DeepStack demonstrated a significant improvement in playing strength compared to previous poker-playing AI agents, and its techniques have inspired further research in the field. One of the key innovations of DeepStack was the use of deep counterfactual value networks to estimate the value of poker hands, enabling efficient real-time decision making.

Recent work by Brown et al. [5] introduced Pluribus, an AI agent that extended the capabilities of Libratus to play at a competitive level against human professionals in 6-player no-limit Texas Hold'em. Pluribus used several novel techniques, including efficient search algorithms and self-play training, to achieve its strong performance in a multiplayer setting. This achievement demonstrates the scalability of AI techniques to larger, more complex poker games and highlights the potential of AI in solving real-world problems with multiple competing agents.

2.2 Reinforcement Learning Techniques

Reinforcement learning has been successfully applied to a variety of games, including board games like Go and chess. One of the most well known examples is AlphaGo by Silver et al. [6], which employed a combination of deep neural networks, Monte Carlo Tree Search (MCTS), and reinforcement learning to defeat the world champion in the game of Go. The success of AlphaGo has led to the development of more advanced AI systems, such as AlphaZero [7], which can learn to play multiple games (Go, chess, and shogi) using a single neural network architecture and a self-play reinforcement learning algorithm.

In the context of poker, Heinrich and Silver [8] introduced Neural Fictitious Self-Play (NFSP), a reinforcement learning algorithm that combines neural networks with fictitious play to learn approximate Nash equilibrium strategies in large-scale poker games. NFSP has been shown to be effective in training agents to play competitive poker, particularly in heads-up limit Texas Hold'em. NFSP leverages the strengths of both deep learning and fictitious play to create a scalable algorithm that can handle the large state and action spaces present in poker.

Other reinforcement learning techniques that have been applied to poker include Counterfactual Regret Minimization (CFR) [9] and Monte Carlo Counterfactual Regret Minimization (MCCFR) [10]. These algorithms have been used to compute near-optimal strategies in various poker variants, demonstrating the potential of reinforcement learning for tackling complex imperfect information games. MCCFR, in particular, has been a key component of several successful poker-playing AI agents, such as Libratus and Pluribus, due to its ability to efficiently approximate optimal strategies through sampling.

The Keras-RL library, which we employ in our project, provides implementations of several reinforcement learning algorithms that can be adapted to poker:

- Deep Q-Network (DQN): DQN [11] is an algorithm that combines Q-learning with deep neural networks to handle high-dimensional state and action spaces. DQN has been successfully applied to numerous Atari games and other environments with discrete action spaces. DQN has been successful in various domains, particularly in environments with discrete action spaces. However, poker involves continuous and large action spaces, which might make it challenging for DQN to excel in this particular setting.
- Deep Deterministic Policy Gradient (DDPG): DDPG [12] is an actor-critic algorithm for continuous control tasks, which utilizes deep neural networks to approximate both the policy and the value function. DDPG is particularly suited for environments with continuous action spaces and has been successfully applied to various robotic control tasks. . Although poker is not a typical continuous control task, DDPG may still perform well due to its ability to handle continuous and large action spaces.
- Cross-Entropy Method (CEM): CEM [13] is a model-free, policy search method that iteratively refines a probability distribution over the solution space to find an optimal policy. CEM has been applied to several optimization problems, including reinforcement learning tasks with discrete and continuous action spaces. Since poker has a large but still discrete action space, this could be advantageous.
- State-Action-Reward-State-Action (SARSA): SARSA [14] is an on-policy, temporal-difference learning algorithm that updates the Q-function after each step in the environment. Its on-policy nature might limit its effectiveness in the complex and dynamic environment of poker, where off-policy learning could potentially be more beneficial.

The OpenAI Gym [15] is a toolkit for developing and comparing reinforcement learning algorithms. It provides a wide range of environments, including classic control tasks, board games, and Atari games, that can be easily integrated with reinforcement learning libraries like Keras-RL. The Clubs package, which we use for simulating poker games, includes a

built-in implementation of the OpenAI Gym interface, allowing for easy integration with Keras-RL agents.

3 Methodology

In this project, we aim to investigate the performance of different reinforcement learning algorithms in the context of 9-player no-limit Texas Hold'em poker. Our approach consists of the following steps:

3.1 Implementing Reinforcement Learning Agents

We utilize the Keras-RL library to implement four reinforcement learning algorithms: Deep Deterministic Policy Gradient (DDPG), Deep Q-Network (DQN), Cross-Entropy Method (CEM), and State-Action-Reward-State-Action (SARSA). Each algorithm is adapted to work with the Clubs_Gym no-limit Hold'em 9-player environment, which provides a realistic and challenging poker simulation.

3.2 Creating Individual Agents

For each of the four reinforcement learning algorithms, we create 9 individual agents, resulting in a total of 36 agents. This setup ensures that in any given poker hand, any combination of agent types playing in the hand is possible. Each type of agent will get experience playing against every combination of opponent over the course of training.

3.3 Training and Evaluation Setup

To train and evaluate the agents, we employ a random selection process in which agents are randomly chosen to play against each other in each hand (episode). This approach forces the agents to adapt to a constantly changing environment and prevents them from learning less meaningful strategies based on the specific opponents they face.

We run a total of 10 million episodes, with each individual agent participating in approximately 278,000 hands on average. Throughout this process, we keep track of the cumulative rewards for each agent.

3.4 Performance Analysis

After completing the training and evaluation process, we analyze the performance of the agents by plotting the cumulative rewards for each agent, as well as the cumulative rewards by agent type. This allows us to assess the overall effectiveness of each reinforcement learning algorithm

and determine if one algorithm consistently outperforms the others in the context of 9-player no-limit Texas Hold'em poker.

By following this methodology, we aim to gain insights into the relative strengths and weaknesses of different reinforcement learning algorithms when applied to a complex, imperfect information game like poker.

4 Results

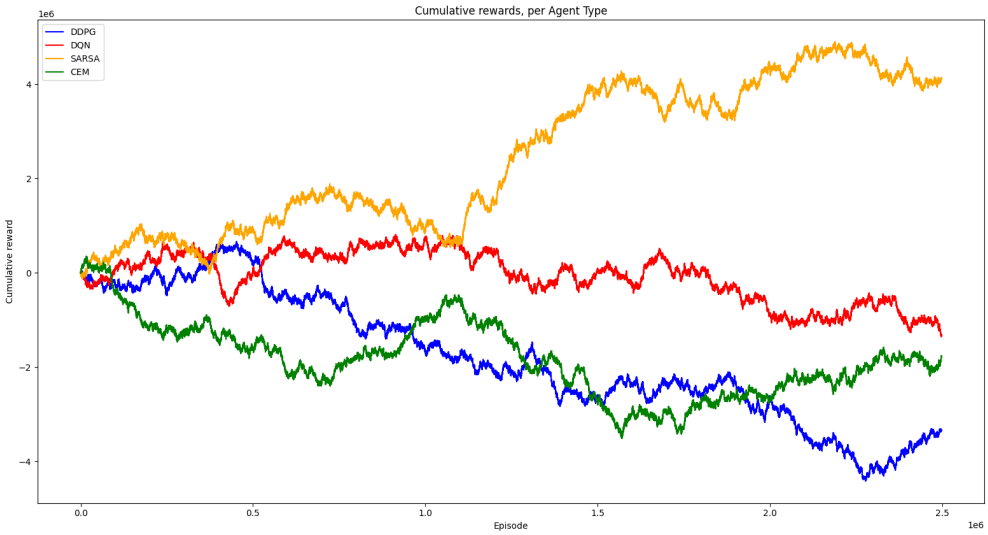

Fig. 1 Cumulative rewards by agent type (SARSA orange, DQN red, CEM green, DDPG blue)

Figures 1, 2, and 3 illustrate how the different reinforcement learning algorithms performed throughout the simulation by plotting their cumulative rewards at each episode. SARSA emerged as the best overall agent type. The top-performing SARSA agent was an outlier, significantly outperforming other all other agents and amassing a substantial total reward.

CEM agents exhibited the least diversity in terms of performance. Their end results were relatively clustered, with none of the agents achieving the highest or lowest rewards.

DDPG agents fared the worst out of the four agent types. Not only did the worst-performing agent belong to this group, but the cumulative reward for DDPG agents was also the lowest.

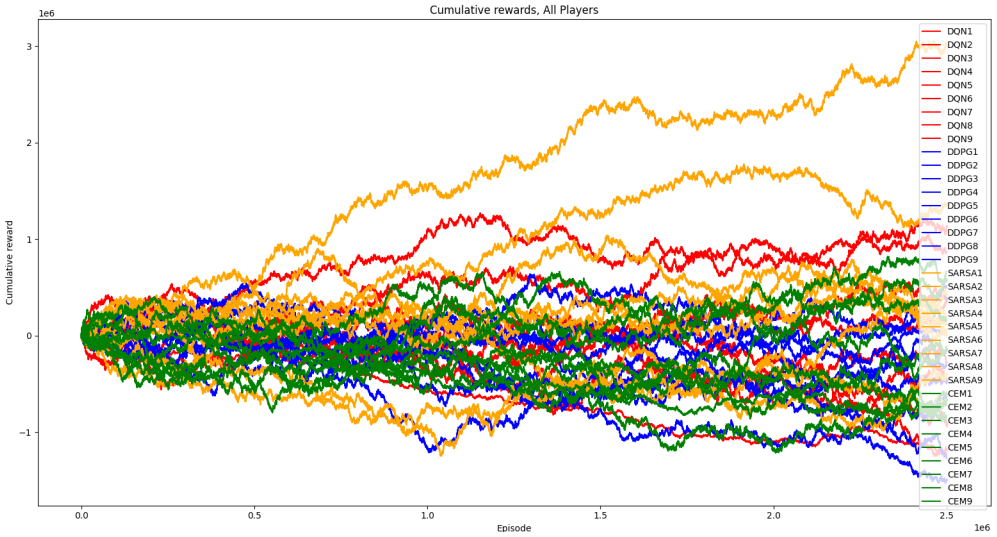

Fig. 2 Cumulative rewards, all agents (SARSA orange, DQN red, CEM green, DDPG blue)

DQN agents displayed a wide range of performance levels, with some agents performing well and others poorly. On average, DQN was the second-best agent type, although it ultimately lost to SARSA.

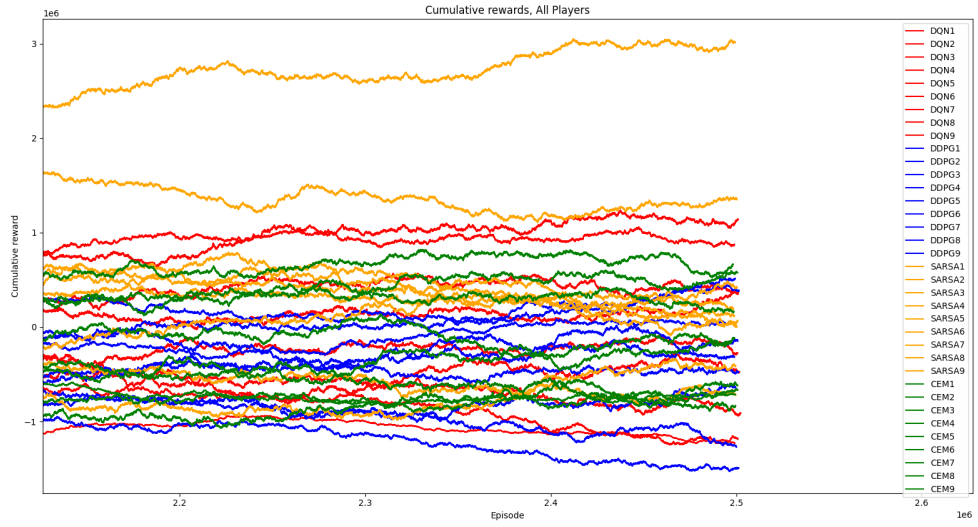

Fig. 3 Cumulative rewards at the end of simulation (SARSA orange, DQN red, CEM green, DDPG blue)

5 Analysis

The superior performance of SARSA agents can be attributed to their on-policy, temporal-difference learning approach, which might have enabled them to adapt more effectively to the dynamic poker environment. The fact that the top-performing SARSA agent was an outlier suggests that specific configurations or training conditions may have contributed to its exceptional performance. It would be interesting to investigate the specific aspects of SARSA's learning process, such as its exploration-exploitation trade-off, that led to its success in this particular setting. Understanding the factors that contributed to the outlier's performance may provide insights into how to improve other SARSA agents and make them more competitive in the poker environment.

The relatively consistent performance of CEM agents could indicate that the policy search method employed by this algorithm is less sensitive to the initial conditions and hyperparameters, resulting in a more stable, albeit not outstanding, performance. This observation raises the question of whether the CEM algorithm's inherent stability can be leveraged to improve its performance through additional tuning or optimization. For instance, incorporating adaptive mechanisms or fine-tuning the exploration process might lead to better results while maintaining the algorithm's consistency.

The poor performance of DDPG agents is somewhat surprising, given their ability to handle continuous and large action spaces. It is possible that the specific implementation details, neural network architectures, or hyperparameters used in our study may have hindered the algorithm's effectiveness in the poker domain. A more detailed examination of the DDPG agents' learning processes could shed light on the challenges they faced and offer insights into potential adjustments that might enhance their performance. Moreover, exploring alternative actor-critic approaches, such as Proximal Policy Optimization (PPO), could provide an interesting avenue for future research in this area.

The diverse performance of DQN agents highlights the algorithm's adaptability in various settings. However, the overall second-best performance of DQN agents suggests that this algorithm may face challenges in dealing with the imperfect information nature of poker, which could impact its ability to learn optimal strategies effectively. Exploring modifications to the DQN algorithm that specifically address the imperfect information aspect of poker, such as incorporating opponent modeling or

exploiting hidden information, might lead to improved performance and better competitive results.

In conclusion, our study demonstrates that SARSA is the most effective reinforcement learning algorithm for 9-player no-limit Texas Hold'em poker among the tested algorithms. However, further research is needed to investigate the factors contributing to the performance of different algorithms and explore potential improvements and optimizations to enhance their effectiveness in the complex and dynamic poker environment. By delving deeper into the nuances of each algorithm and their learning processes, we can develop a better understanding of the challenges and opportunities in applying reinforcement learning to imperfect information games like poker.

6 Conclusion

In this study, we investigated the performance of four reinforcement learning algorithms, namely SARSA, DQN, CEM, and DDPG, in the context of 9-player no-limit Texas Hold'em poker. Our results indicate that SARSA was the most effective algorithm among the tested methods, while DQN ranked second, followed by CEM and DDPG. This outcome highlights the potential of reinforcement learning techniques, particularly SARSA, in tackling complex imperfect information games like poker.

Despite the promising results, our study also revealed certain limitations and challenges faced by the tested algorithms. These findings emphasize the need for further research and optimizations to improve the performance of reinforcement learning agents in poker and other imperfect information domains. Moreover, our study opens up several avenues for future work:

Investigating the specific factors contributing to the top-performing SARSA agent's success and exploring methods to replicate this performance in other SARSA agents. Examining the stability of CEM agents and exploring potential improvements through adaptive mechanisms or fine-tuning the exploration process. Analyzing the reasons behind the poor performance of DDPG agents in the poker domain and evaluating alternative actor-critic approaches, such as Proximal Policy Optimization (PPO). Exploring modifications to the DQN algorithm that address the imperfect information nature of poker, such as incorporating opponent modeling or exploiting hidden information. By pursuing these research directions, we can gain a deeper understanding of the challenges and

opportunities in applying reinforcement learning to imperfect information games and contribute to the ongoing development of more effective and adaptable AI agents in poker and beyond.

References

[1] Keras-rl: Deep Reinforcement Learning for Keras. GitHub. https://github.com/keras-rl/keras-rl (2019)

[2] clubs_gym: Open AI gym poker environment built using the clubs package. GitHub. https://github.com/fschlatt/clubs_gym (2022)

[3] Brown, N., Sandholm, T.: Libratus: The superhuman ai for no-limit poker. In: International Joint Conference on Artificial Intelligence (2017)

[4] Moravč ík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., Bowling, M.: DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science **356**(6337), 508–513 (2017) https://doi.org/10.1126/science.aam6960

[5] Brown, N., Sandholm, T.: Superhuman ai for multiplayer poker. Science **365**, 885–890 (2019)

[6] Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–489 (2016) https://doi.org/10.1038/nature16961

[7] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (2017)

[8] Heinrich, J., Silver, D.: Deep Reinforcement Learning from Self-Play in Imperfect-Information Games (2016)

[9] Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Advances in Neural Information Processing Systems, pp. 1729–1736 (2007)

[10] Lanctot, M., Waugh, K., Zinkevich, M., Bowling, M.: Monte carlo sampling for regret minimization in extensive games. In: Advances in Neural Information Processing Systems, pp. 1078–1086 (2009)

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra,

D., Riedmiller, M.A.: Playing atari with deep reinforcement learning. CoRR **abs/1312.5602** (2013) 1312.5602

[12] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015)

[13] Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Springer, ??? (2004)

[14] Rummery, G.A., Niranjan, M.: On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department (1994)

[15] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016)