Foundations of Artificial Intelligence: Final Project Report

b09902060 馮楷

1 Introduction

For this final project, which focuses on developing an Agent capable of playing Texas Hold'em poker, I employed three different approaches. These approaches encompassed (1)mimicking human decision-making and thought processes through an expert system that explicitly defined all rules, (2)utilizing a static model based on Monte Carlo algorithms to calculate the current equity and make decisions accordingly and (3)training a machine learning model using Reinforcement Learning. Each of these three approaches has its own advantages and disadvantages. Consequently, I also explored combining different models to examine whether the performance of a fused model surpasses that of individual methods.

2 Approaches

2.1 Expert System

As an avid Texas Hold'em player, my initial approach to this project was to attempt to program my own strategies and thought processes.

Firstly, although the game appears to be played with a stack of 100 big blinds (BB), it can be theoretically simplified to a 15 BB game due to the maximum number of rounds. In this case, a preflop action table with a positive expected value can be referenced. Thus, I constructed a model based on such action tables to determine my preflop actions. Additionally, I kept track of my own actions and my opponents' actions on each street, whether they were playing aggressively or defensively, and made decisions accordingly. For example, if the opponent showed weakness (checked) on two consecutive streets, I increased my betting frequency. On the other hand, if the opponent chose to attack (raise) on multiple streets, I would consider folding more hands.

Furthermore, I observed that opponents' betting tendencies were relatively linear. Therefore, I could exploit this by removing many preflop aggressive actions (such as semi-bluffing with blockers) to attack the opponents. In other words, in the event of a preflop all-in situation, most of the time my hand would be ahead a lot.

2.2 Monte Carlo-based Model

The core concept of the Monte Carlo-based model is to calculate the current hand's equity within a given range by randomly assigning opponent hands and community cards multiple times. This calculated equity serves as the basis for making decisions. A possible example could be: if equity ¿ 0.9, the decision is to go all-in; if equity ¿ 0.6, the decision is to raise to a pot size; if equity ¿ 0.4, the decision is to call; otherwise, fold. Based on this concept, I implemented two different random ranges:

1. **All unknown cards.** The most naive approach to estimating the equity is to randomly assign the remaining unknown cards to opponent's hand and community cards. This allows an analysis of the current hand against all possible scenarios that may occur.

2. **All reasonable hand combinations.** Just as no rational player would go all-in with 27o, opponents' hands that can call or raise should fall within a reasonable range (e.g., the top 75%/50% of hands, respectively). Therefore, at the beginning of each street, I inferred the opponent's hand range based on their previous actions and analyzed the current hand's equity against this range, providing a theoretically more accurate estimation.

2.3 Reinforcement Learning Model

Training a model using Reinforcement Learning (RL) has its limitations, such as the need for a closed environment for testing and a clear way to determine wins and losses through a reward function. Texas Hold'em poker, however, satisfies these requirements pretty well. The model I chose to employ is Neural Fictitious Self-Play [2], where each hand is treated as a unit, and an evaluation is performed every 100 hands. In total, the model was trained over 10,000 hands. The opponents in this training setup consist of the two previously implemented agents mentioned earlier, i.e., the Expert System and the Monte Carlo-base Model, along with a Random Agent.

3 Comparison & Discussion

3.1 Expert System

The limitations of the expert system approach are quite evident. Firstly, regarding the assumption of simplifying the game to 15 BB, it is apparent that all baselines (except baseline2) do not strictly consider the game as a 15 BB game. Consequently, the decisions derived from this assumption, whether in terms of own betting scale, betting/calling ranges, or estimating opponent ranges, lack significant reference value.

Furthermore, the expert system relies on complex, explicitly defined rules. However, it fails to make good decisions in situations that were not specifically considered during programming. Additionally, it is nearly impossible for the expert system to make adjustments for different opponents, which results in it performing the worst among all the models. When facing baseline1 and baseline2, the model only achieves a winning rate of slightly more than 50%.

3.2 Monte Carlo-based Model

Surprisingly, the second method (all reasonable hand combinations), which theoretically should be more accurate, performed worse in actual testing compared to the first method (all unknown cards). The reason behind this is that the second method heavily relies on the accuracy of estimating opponent ranges. When the estimated range deviates significantly from the actual range, the subsequent calculated equity become distorted. Furthermore, this model lacks the ability to adjust itself, so even if it happens to guess the range of a particular opponent correctly, it would be helpless when facing opponents with completely different ranges.

Despite this, its performance is still better than the expert system, even with the inferior second method (all reasonable hand combinations). It achieves a winning rate of over 70% when facing baseline1 and baseline2.

3.3 Reinforcement Learning Model

The model trained directly against the baseline performs exceptionally well; however, this approach is not feasible. As for the models trained using expert systems and other methods, their performance is not as outstanding as anticipated and is generally comparable to the first-type Monte Carlo-based model. One possible reasons for this includes differences in training settings and actual scenarios, as the training did not include the constraint of determining a winner every 20 rounds.

Additionally, it is possible that the performance of the trained model is not as expected due to significant differences in the behavior of the opponent models compared to the baseline. These differences in behavior could impact the learning outcome adversely.

4 Conclusion

4.1 Final Result

Due to the challenges of combining the RL model with the other two methods, I ultimately chose to combine the expert system and the Monte Carlo approach to improve performance.

I retained the preflop table but modified the strategy to be more linear. I also introduced more flexibility by converting some hands that would previously go all-in directly into equivalent raises of 15 BB. Additionally, the criteria for postflop actions shifted from solely considering hand strength to a combination of strength and equity. Furthermore, I adjusted the strategy for being behind in chips, becoming more aggressive and increasing the betting scale after the 10th round if not in a leading position.

In the end, this combination model demonstrated the best performance. Not only did it achieve a winning rate of over 80% against the first three baselines, but it also maintained a winning rate of over 50% when facing baseline4 and baseline5.

4.2 Conclusion

In this final project, I explored three distinctly different approaches and discussed the advantages and disadvantages of each method. Ultimately, while preserving the unique characteristics of each approach, I combined the expert system with the Monte Carlo method and created a well-performing hybrid model.

References

[1] Ishikota. (2017, April 2). *PyPokerEngine*. GitHub. https://github.com/ishikota/PyPokerEngine

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

[3] Lai, H. (2019, November 12). *RLCard: Building Your Own Poker AI in 3 Steps*. Medium. https://towardsdatascience.com/rlcard-building-your-own-poker-ai-in-3-steps-398aa864a0db