FAI FINAL PROJECT Report

b11705051 CHEN YI-TING

2024/06/15

1 Introduction

The aim of this project was to develop an AI player for a poker game using Q-learning. Various methods and configurations were explored to determine the most effective strategy for the AI player.

2 Methods Tried

2.1 Method 1: Basic Q-Learning with Card States

- **Description**: The initial approach used the card states (hole cards and community cards) directly as part of the state representation.
- **State Representation**: String concatenation of hole cards and community cards.
- **Reward**: Based on the strength of the final hand.
- **Results**: This method had a large state space, making learning slow and inefficient. It also required a lot of space to store all the combinations and was not very general.

2.2 Method 2: Q-Learning with Simplified State (Win Rate)

- **Description**: To reduce the state space, the state was represented using the computed win rate instead of card states.
- **State Representation**: The state is represented as a string in the format `xx | xx | xx`, where each segment corresponds to the win rate calculated at each round. The win rate obtained in each round is appended to the end of the string, providing a cumulative representation of the win rates across rounds. This method allows the AI to consider the sequence of win rates throughout the game, enhancing its decision-making process.
- **Reward**: Based on the strength of the final hand.

- **Results**: It required less space and improved learning efficiency and better performance compared to the first method.

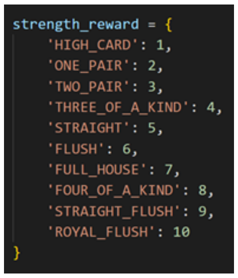
2.3 Method 3: Q-Learning with Domain Knowledge

- **Description**: For the above two methods, the model was allowed to choose a random step to explore more. However, this did not work very well. Thus, I decide to add some domain knowledge to avoid bad behaviors, like raising when win rate is a 0. Also I won't let the machine randomly choose the decision, I do decision making base on the win rate.
- **State Representation**: As method 2.
- **Action Space**: `['fold', 'call', 'raise']`
- **Reward**: Based on the strength of the final hand and game outcome.
- **Results**: It has improved significantly. While Method 2 still cannot consistently beat the weaker baselines, it has managed to narrow the gap in losses. Although the machine has not yet achieved consistent victories against all baselines, it shows promise in reducing the margin of defeat.

3 Configurations (Hyperparameters)

3.1 Hyperparameters Used

- Learning Rate (α): 0.05
- Discount Factor (γ): 0.9
- Exploration Rate (ϵ): 1.0 (initial)
- Exploration Decay (ϵ_{decay}): 0.99
- Reward Structure:



```
strength_reward = {
    'HIGH_CARD': 1,
    'ONE_PAIR': 2,
    'TWO_PAIR': 3,
    'THREE_OF_A_KIND': 4,
    'STRAIGHT': 5,
    'FLUSH': 6,
    'FULL_HOUSE': 7,
    'FOUR_OF_A_KIND': 8,
    'STRAIGHT_FLUSH': 9,
    'ROYAL_FLUSH': 10
}
```

Figure 1: strength reward

3.2 Final Configuration

The final chosen configuration used the Method 3: Q-Learning with Domain Knowledge. This configuration achieved a good balance between learning efficiency and performance. It would also adjust the bad habits machine trying to make.

4 Comparison of Methods

4.1 Comparison Metrics

- Learning Efficiency: Speed at which the AI improved its performance.
- Performance: Win rate and stability in decision-making.
- State Space Size: Complexity of the state representation.

4.2 Results

- **Method 1**: High state space complexity, slow learning, lower performance.
- **Method 2**: Reduced state space, faster learning, improved performance.
- **Method 3**: Balanced exploration and exploitation, efficient learning, stable and better performance.

5 Discussion

In the methods discussed above, Method 3 has demonstrated the best performance. However, given the constraints of my current computational resources, there is a concern that Method 3 may have reached a local maximum rather than a global maximum. This suggests that with improved training methods or more powerful computing resources, Method 2 could potentially be leveraged to train a larger model without relying on domain-specific knowledge, allowing the machine to learn the entire game dynamics more comprehensively.

6 Conclusion

In this project, we explored various approaches to develop an AI player for a poker game using Q-learning. Three main methods were tested, each refining the state representation and decision-making process of the AI.

Method 1 utilized direct card states, leading to a large state space and slower learning. Method 2 simplified the state representation to win rates, which improved efficiency and performance. Finally, Method 3 incorporated domain knowledge to refine decision-making, resulting in significant performance gains and stability.

Through comparative analysis, Method 3 emerged as the most effective approach, balancing exploration and exploitation to achieve efficient learning and robust performance. Despite its successes, limitations in computational resources suggest that Method 2 could potentially offer a broader exploration of game dynamics with increased training capacity.

Looking forward, further enhancements could involve refining reward structures or exploring more sophisticated learning algorithms beyond Q-learning. These efforts could unlock additional improvements and potentially achieve higher levels of gameplay proficiency.

Overall, this project underscores the effectiveness of Q-learning in developing AI strategies for complex games like poker. The insights gained contribute to the ongoing evolution of AI techniques in gaming and decision-making domains.