FAI Final Project Report

B10201051 Yu-Ling Liao

The methods I have tried

I tried two methods to develop my agent.

The first method is very straightforward. I used if-else statements to determine the agent's actions, dividing the process into preflop and postflop stages. In the preflop stage, I categorized the strength of the starting hands in detail. Then, determined the appropriate action based on the hand strength and player position. For example, if I received strong hole cards, I would raise regardless. If the hole cards were of medium strength, I would decide to call or fold based on my current position. If the hole cards were weak, I would choose to fold directly. In the postflop stage, I used an encoding method found on internet to calculate the strength of my hole cards combined with the community cards, and I decided on actions based on the estimated return rate.

Since the first method is too straightforward and there are many aspects I haven't considered (such as the opponent's behavior and the current call amount, etc.), I thought I should use what I learned in class to develop a trained agent from a machine learning perspective. After searching online, I decided to use Q-learning from reinforcement learning to develop this agent. With the help of a GitHub repo shared by classmates in the discussion, I added emulator, card_utils, and game_state_utils to assist in development. I also used the existing random_player to help train the agent.

Configurations

MyEnhancedPokerPlayer

1. Initialization: UUID-Player identifier.
2. Action Declaration: Uses different strategies for preflop and postflop stages.
3. Preflop Strategy: Determines action based on hand rank and player position.
4. Position Calculation: Identifies player position relative to the dealer button.
5. Hand Evaluation:Ranks hands using a custom scoring system for preflop.

RLPlayer

1. Initialization:
    - Q-Table: A dictionary to store state-action values.
    - Learning Parameters:
        - learning_rate (α): 0.1
        - discount_factor (γ): 0.9
        - exploration_rate (ϵ): 1.0 with decay of 0.99
    - Emulator: Used for simulating game conditions.
2. Game Start Configuration: Registers players and sets game rules using the emulator.

3. Action Declaration:

   - Uses ϵ-greedy strategy: Random action with probability ϵ. Best action based on Q-values otherwise.
   - Round Result Handling:
     - Updates Q-values based on received reward and the new state.
     - Decays the exploration rate.
     - State Representation: Combines hole cards, current street, and pot amount.
     - Q-Table Update: $Q(s,a) \leftarrow Q(s,a) + \alpha\,(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$

Comparison of the methods

Key Differences

MyEnhancedPokerPlayer employs predefined strategies for different states of the game, while **RLPlayer** uses strong q-learning method to dynamically learn the best actions over time, which is adaptable and improves with more game as it learns from outcomes. On the other hand, **MyEnhancedPokerPlayer** more focuses on the hole card and current position, which might ignore some important information such as the opponent's action. **RLPlayer** is more generally to consider different state representation, so that it can facing more different situation.

MyEnhancedPokerPlayer v.s. RLPLayer

```
Game1: The winner is RLPLayer with a remaining stack of 2000
Game2: The winner is RLPLayer with a remaining stack of 1045
Game3: The winner is RLPLayer with a remaining stack of 1030
Game4: The winner is MyEnhancedPokerPlayer with a remaining stack of 1965
Game5: The winner is RLPLayer with a remaining stack of 2000
```

Figure 1: RLPlayer win the BO5

MyEnhancedPokerPlayer v.s. baseline0_ai

```
Game1: The winner is MyEnhancedPokerPlayer with a remaining stack of 2000
Game2: The winner is MyEnhancedPokerPlayer with a remaining stack of 2000
Game3: The winner is MyEnhancedPokerPlayer with a remaining stack of 2000
Game4: The winner is baseline0_ai with a remaining stack of 1150
Game5: The winner is baseline0_ai with a remaining stack of 2000
```

Figure 2: MyEnhancedPokerPlayer win the BO5

RLPlayer v.s. baseline0_ai

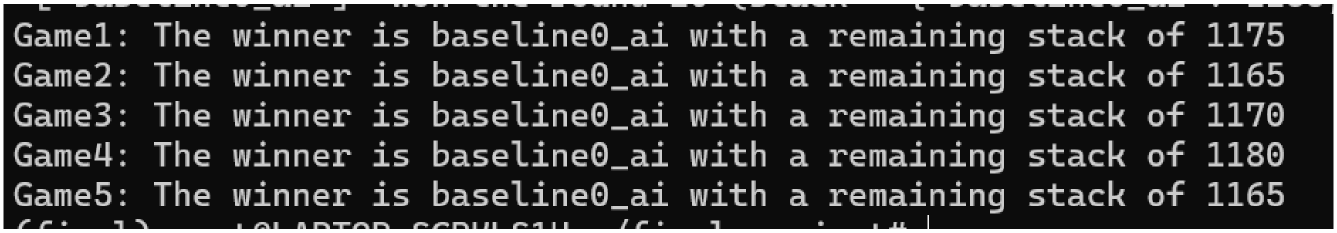

Figure 3: baseline0_ai win the BO5

Discussion and Conclusion

- **RLPLayer:**

  Pros: Learns and adapts over time, potentially outperforming static strategies in the long run. Suitable for environments where long-term adaptation and learning are crucial.

  Cons: More complex to implement and requires a large number of games to converge to optimal strategies, may result int bad performance sometimes since not complete implementation:(

- **MyEnhancedPokerPlayer:**

  Pros: Simpler to implement and faster in decision-making. Effective if the predefined strategies are robust and tailored to specific game conditions. Aggression can sometimes have very good consequences.

  Cons: Lacks adaptability and may perform poorly against sophisticated opponents or in dynamic game environments.

Method I choose to submit finally

RLPlayer should be the better agent to submit, but unfortunately, perhaps due to my insufficient implementation skills, even though I spent a lot of time training the agent, it did not perform very well when competing against baselines_ai. Despite further optimization with online searches and GPT assistance, the agent's performance was still inferior to the former. Due to time constraints, I decided to submit the first agent, **MyEnhancedPokerPlayer**, for the competition, as it wins against baselines_ai more often than RLPlayer. However, I have also included my implementation of the second agent.

Reference

PyPokerEngine github repo:
https://github.com/ishikota/PyPokerEngine &
https://ishikota.github.io/PyPokerEngine/tutorial/simulate_the_game_by_emulator/
Hole card and all card evaluation:
https://www.pokerstrategy.com/poker-hand-charts-evaluations/ &
https://cowboyprogramming.com/2007/01/04/programming-poker-ai/
Q-learning and Deep-Q-Network:
https://www.adaltas.com/en/2019/01/09/applying-deep-reinforcement-learning-poker/ &
https://medium.com/@mycorino/building-a-deep-q-network-powered-poker-bot-1a48e296805d