

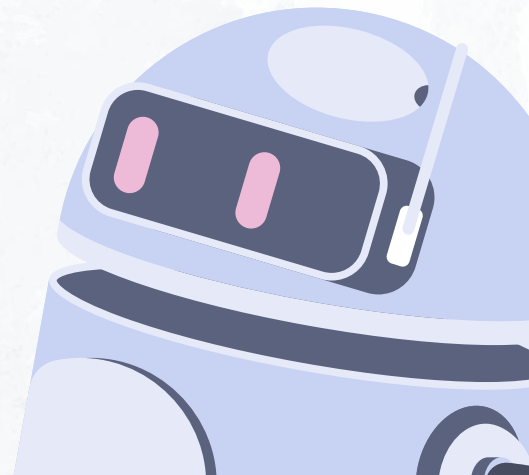
FAI Final Project

Texas Hold'em Casino



FAI 2023 Spring Semester
Instructors: Shang-Tse Chen & Yun-Nung Chen
Due date: 2023/06/21 23:59

(AI)



Outline

- 01 → Project Introduction and Rules
- 02 → Game Parameters & Environment in Final Project
- 03 → Sample Code Explanation
- 04 → Competition Rules
- 05 → Submission
- 06 → Grading Policy

01



Introduction

Implement Texas Hold'em AI to compete
in the CSIE casino



(AI)

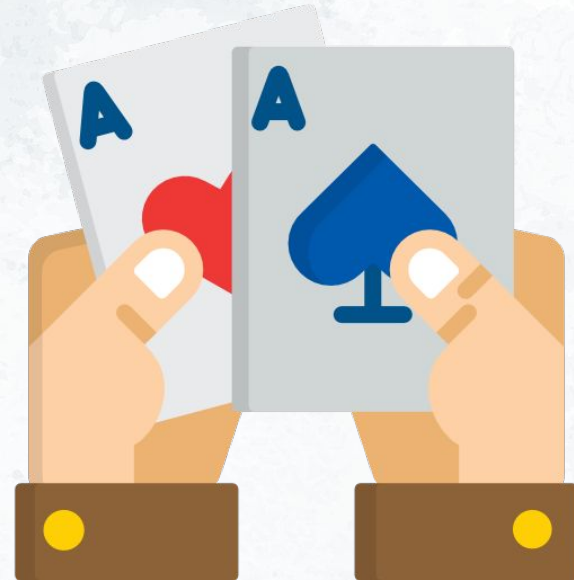
Texas Hold'em Rules –Heads Up

- Each player has two face-down cards, called the “**hole cards**”
- The face-up cards are called the “**community cards**”
- Goal: use the community cards combined with their hole cards to build a five-card poker hand (Build best 5 combination out of 7 cards)



Small Blind & Big Blind

- A game features **several betting rounds**
- Players get two private and up to five community cards
- All blinds **must** bet
 - 1st player: “small blind”
 - 2nd player: “big blind”
 - Big blind is two times larger than small blind

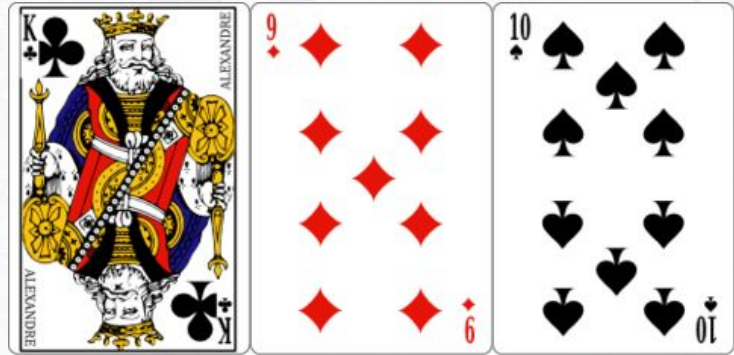


First Betting Round: Preflop

- The player has 3 options
 - **Call:** match the amount of the big blind
 - **Raise:** increase the bet within the limits of the game
 - **Fold:** throw the hand away. If the player chooses to fold, he or she is out of the game and no longer eligible to win the current hand.

Second Betting Round: Flop

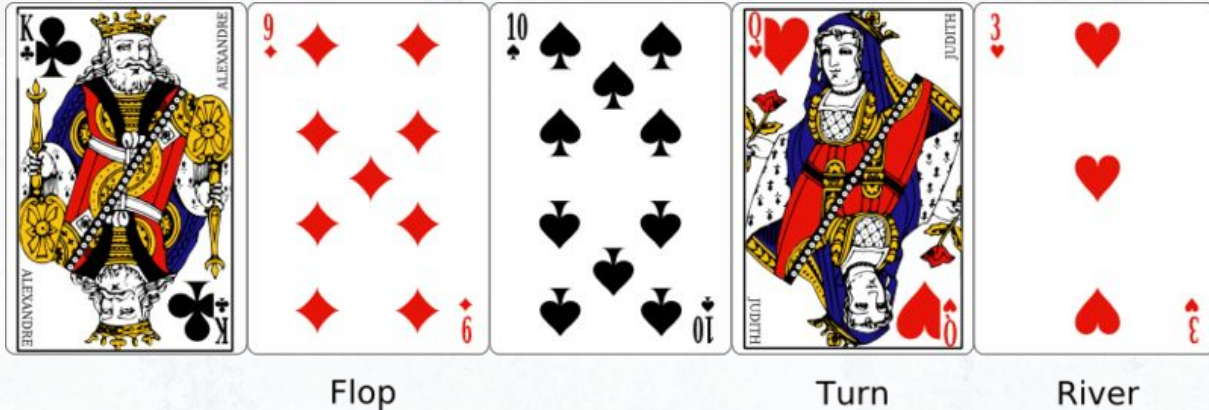
- The first three community cards are dealt
- Players can only choose action **“call”**, **“raise”** or **“fold”**



Flop

Third & Fourth Betting Round: Turn & River

- The fourth community card, called the **turn**, is dealt face-up
- Players can only choose action “call”, “raise” or “fold”
- The fifth community card, called the **river**, is dealt face-up
- Players can only choose action “call”, “raise” or “fold”























Showdown






























- All players expose their holdings to determine a winner.



Hands Rank

- **Royal Flush** — five cards of the same suit, ranked ace through ten; e.g.,   
 
- **Straight Flush** — five cards of the same suit and consecutively ranked; e.g.,  
  
- **Four of a Kind** — four cards of the same rank; e.g.,     
- **Full House** — three cards of the same rank and two more cards of the same rank; e.g.,     

Hands Rank

- **Flush** — any five cards of the same suit; e.g.,     
- **Straight** — any five cards consecutively ranked; e.g.,     
- **Three of a Kind** — three cards of the same rank; e.g.,     
- **Two Pair** — two cards of the same rank and two more cards of the same rank; e.g.,    
- **One Pair** — two cards of the same rank; e.g.,     
- **High Card** — five unmatched cards; e.g.,      would be called "ace-high"

02 →

Game Parameters & Environment

(AI)

Game Parameters in Final Project

- The winner is the one has more money after all games
 - Max round of game: 20
 - Initial stack for each player: 1000
 - Small blind: 5
- No upper bound for “raise”, you can **all in** at once.
- You must take action within **5** seconds in every turn, or treated as “**fold**”
- You must take a valid action in every turn, or treated as “**fold**”

Environment

- Setup python version, on *linux{1~15}.csie.org* workstation

```
pyenv install -v 3.8.13
```

- Switch to the installed python 3.8.13

```
pyenv global 3.8.13
```

- Install the packages

```
pyenv exec pip install -r requirement.txt
```

- Run the ***start_game.py*** to see how the game works

```
pyenv exec python start_game.py
```

Environment

- python 3.8.13
- numpy==1.22.3
- torch==1.11.0
- scikit-learn==1.0.2
- tensorflow==2.8.0
- keras==2.8.0
- pytorch_lightning==1.6.1
- tqdm==4.64.0
- If you want to use other packages, please email the TA first.

03 →

Sample Code Explanation

(AI)

Sample Code Explanation

- `final_project/`
 - `|--- start_game.py` *<the code to help you test locally>*
 - `|--- requirement.txt` *<python package needed in this project>*
 - `|--- baseline0.cpython-38-x86_64-linux-gnu.so`
<the binary file that can be import in csie workstation>
 - `|--- game/`
<it contains all needed game objects, you should not modify any file in this directory>
 - `|--- agents/` *<it contains sample agents for you to play with>*
 - `|--- call_player.py` *<the agent always to “call” action>*
 - `|--- random_player.py` *<the agent choose action randomly>*
 - `|--- console_player.py` *<you can play the game in interaction mode with this player>*

Sample Code Explanation

- Your agent should make parent class as **“BasePokerPlayer”**
- You should override 7 functions
 - declare_action
 - receive_game_start_message
 - receive_round_start_message
 - receive_street_start_message
 - receive_game_update_message
 - Receive_round_result_message
- You should include a function named **“setup_ai”** that return your agent class

```
1 from game.players import BasePokerPlayer
2
3 class CallPlayer(BasePokerPlayer): # Do not forget to make parent class as "BasePokerPlay
4
5     # we define the logic to make an action through this method. (so this method would be
6     the core of your AI)
7     def declare_action(self, valid_actions, hole_card, round_state):
8         # valid_actions format => [raise_action_info, call_action_info, fold_action_info]
9         call_action_info = valid_actions[1]
10        action, amount = call_action_info["action"], call_action_info["amount"]
11        return action, amount # action returned here is sent to the poker engine
12
13    def receive_game_start_message(self, game_info):
14        pass
15
16    def receive_round_start_message(self, round_count, hole_card, seats):
17        pass
18
19    def receive_street_start_message(self, street, round_state):
20        pass
21
22    def receive_game_update_message(self, action, round_state):
23        pass
24
25    def receive_round_result_message(self, winners, hand_info, round_state):
26        pass
27
28    def setup_ai():
29        return CallPlayer()
```

Sample Code Explanation

- To test your agent locally, you can use *"python3 start_game.py"*
 1. Import setup_ai function for every agent
 2. Setup game configuration with predefined rules
 3. Register users with the agent
 4. Play the game and get the result

```
1 from game.game import setup_config, start_poker
2 from agents.call_player import setup_ai as call_ai
3 from agents.random_player import setup_ai as random_ai
4 from agents.console_player import setup_ai as console_ai
5
6 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
7 config.register_player(name="p1", algorithm=call_ai())
8 config.register_player(name="p2", algorithm=random_ai())
9 config.register_player(name="me", algorithm=console_ai())
10 game_result = start_poker(config, verbose=1)
```


Sample Code Explanation

- To play the game interactively, you can use "console_player.py".
 - It allows you to play the game step by step.
 - Baselines cannot be played in an interactive mode; you can only test it locally.

```
1 from game.game import setup_config, start_poker
2 from agents.call_player import setup_ai as call_ai
3 from agents.random_player import setup_ai as random_ai
4 from agents.console_player import setup_ai as console_ai
5
6 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
7 config.register_player(name="p1", algorithm=call_ai())
8 config.register_player(name="p2", algorithm=random_ai())
9 config.register_player(name="me", algorithm=console_ai())
10 game_result = start_poker(config, verbose=1)
```


Sample Code Explanation

- Example result after submit to baseline server.
- Game Rules
- Game result after playing 20 runs
- p1 won in this example

```
{
  "rule": {
    "initial_stack": 1000,
    "max_round": 20,
    "small_blind_amount": 5,
    "ante": 0,
    "blind_structure": {}
  },
  "players": [
    {
      "name": "p1",
      "uuid": "jwda:mmhainrbmufgfgodc",
      "stack": 1996,
      "state": "participating"
    },
    {
      "name": "p2",
      "uuid": "ikscmdgatlxjeozsrhhbsd",
      "stack": 0,
      "state": "folded"
    }
  ]
}
```

Sample Code Explanation

- We will be given **5 black-box baselines** (different levels of difficulty)
- Baselines are in **binary executable format** compatible in csie workstation
 - <your student id>@linux{1~15}.csie.org
 - If you don't have a csie workstation account, please refer to the [link](#)
- We will release baseline agents in the following dates.
 - baseline0 (test only;no credit): 5/21
 - baseline1 & baseline2: 5/24
 - baseline3 & baseline4: 5/31
 - baseline5: 6/07

```
6 from game.game import setup_config, start_poker
7 from agents.random_player import setup_ai as random_ai
8 from baseline0 import setup_ai as baseline0_ai
9
10 config = setup_config(max_round=20, initial_stack=1000, small_blind_amount=5)
11 config.register_player(name="p1", algorithm=baseline0_ai())
```

04 →

Competition Rules

(AI)

Competition Rules

- Each match (two players compete with each other) consists of 5 games
- Each game consists of 20 rounds starting from 1,000 stacks for both
- The player “**wins**” the game if they have **more money left** after 20 rounds

Competition Rules

- **Round-Robin Tournament**

- Each player is randomly assigned into **a group of 6 people**
 - 5 opponents in total
 - Winners for the matches will get 2 points
 - You can get maximum 10 points if you win all matches

- **Single Elimination**

- Top 32 players with highest scores (left money summed over 5 matches) are selected to participate in the single elimination tournament
- Top 4 can get bonus for the final grades (4pt~1pt)

05 →

Submission

(AI)

Submission

- Due date: 2023/06/21 23:59
- Upload to NTU COOL
- No late submission is allowed

Submission

- TAs will evaluate both **code functionality** and the **report quality**
- Please compress your agent and the related files in a single .zip named with your student id in lowercase.
- The size of the submitted .zip file should be less than 500MB.
- Example:
 - b0x902xxx.zip
 - b0x902xxx/
 - |-- report.pdf
 - |-- src/
 - |-- agent.py
 - |-- other related file needed

Report

- Your report should include but not limited to
 - The methods you have tried
 - Your configurations (e.g. hyperparameters)
 - Comparison of your methods
 - Discussion and conclusion
 - What method you choose to submit finally.
- You should write your report in maximum **4 A4 pages in pdf format**
- The grading of the report will base on the **number of methods you tried, completeness, novelty, and clarity of writing**

06 →

Grading Policy

(AI)

Grading (105 pts in total)

- Report (55 pts)
- Baseline Competition (30 pts)
 - baseline1~baseline5 (6 pts for each)
- **Unseen** Strong Agent Competition (10 pts)
 - strong1~strong5 (2 pts for each)
- Round-Robin Tournament (10 pts)
 - Total 5 matches (2 pts for each)
- Single Elimination Tournament (bonus 4 pts)

fai23@csie.ntu.edu.tw

Q & A

