Poker Project

This is the two-player no-limit Texas hold'em poker competition project.

You only need to implement your own poker agent client. Although the poker server is implemented in C++, we do not have any restrictions on the language with which your poker agent is built on. You can choose C/C++, python, java, etc. to build client executables, as long as it satisfies the following requirements:

- It takes at least two arguments:
  - 1st argument: address of the server. E.g., 127.0.0.1 for a localhost server.
  - 2nd argument: port with which to communicate with the server.
- It safisfies the desiring communication protocol. Please refer to **"POKER - Protocol Specification"** for details if you want to understand messages sent between clients and server, or if you are going to implement your own client from scratch.

For C++ & Python3 users, we have already implemented all components needed for communication with server. You only need to implement your own poker strategy in `act` function in `example_player.cpp` or `example_player.py`. Please refer to **"POKER - example_player"** doc for more details.

For others, you have to build your own client including the communication part. In this case, our implementation in C++ & Python3 can be a useful reference.

Requirements

This server was developed and tested for use on Unix based systems. You will need standard Unix developer tools to build the software including gcc, and make. You will need Python3 if you would like to use `example_player.py`.

Getting Started

Building

The Makefile provides instructions for compiling the code.

- `make`: compile all required programs.
- `make clean`: remove compiled objects.
- `make clean_log`: remove all match logs under `logs` directory.

The programs

- `dealer`: Hosts a poker game, including dealing cards, asking corresponding player to act, etc. The dealer communicates with poker agents over sockets.
- `example_player.cpp`: A *sample* poker player implemented in C++. Communicates with the dealer over sockets. You NEED to implement your own poker agent.

- `example_player.py`: A *sample* poker player implemented in Python3. Communicates with the dealer over sockets. You NEED to implement your own poker agent.
- `play_match.pl`: A perl script for running matches with the dealer.

Playing a match

The fastest way to start a match is through the play_match.pl script. An example follows:

```
$ ./play_match.pl matchName holdem.nolimit.2p.reverse_blinds.game 1000 0 Alice
./example_player Bob ./example_player_python.sh
```

- `matchName`: You can give an arbitrary name to the match.
- `holdem.limit.2p.reverse_blinds.game`: Configurations of two-player no-limit Texas hold'em poker rules. DO NOT CHANGE THIS.
- `1000`: An integer, number of hands to play. 1000 means playing poker for 1000 hands and calculates overall gain or loss.
- `0`: Random seed.
- `Alice` and `Bob`: You can give arbitrary names for the two players.
- `./example_player` & `./example_player_python.sh`: Your poker agents for corresponding players. In this example, we use C++ implementation of `example_player.cpp` as Alice, and Python3 implementation of `example_player.py` as Bob.

When `play_match.pl` finishes, there will be two output files for the dealer and two output files for each player under `logs` directory. The information can be useful for debugging, showing how your agents actually played at each state.

- `matchName.err`: The stderr from dealer including the messages sent to players. Please refer to **"POKER - Protocol Specification"** if you would like to know details about the message format.
- `matchName.log`: The log for the hands played during the match, one line for each hand. Each line consists of "betting sequences", "private and public cards", "wining or losing money", and "agent names".
- `matchName.playerN.std`: stdout from player N.
- `matchName.playerN.err`: stderr from player N.

Another approach

However, if you have problems running the perl script, or you would like to debug your poker client, it is more convenient to run each part separately.

First, start a `dealer` server by running

```
$ ./dealer matchName holdem.nolimit.2p.reverse_blinds.game 1000 0 Alice Bob
58669 58670
# name/game/hands/seed matchName holdem.nolimit.2p.reverse_blinds.game 1 0
#--t_response 600000
#--t_hand 600000
#--t_per_hand 7000
```

The arguments are the same with the previous approach. The server prints two port numbers, 58669 and 58670. Each port needs to be connected with one poker client. Now we start two poker clients (e.g. example_poker). You NEED to replace port number with the one printed by your dearler server.

```
$ ./example_player 127.0.0.1 58669
```

```
$ ./example_player 127.0.0.1 58670
```

Here, 127.0.0.1 is the localhost address of our server. You may use any kind of executable poker clients which satisfies requirements described at the beginning of the document.

After the two poker clients are running and connected, the game begins.