Bachelor Thesis - Poker Simulator

Study Material

Fabian Moik

Inhaltsverzeichnis

1 Poker Simulator

1.1 Basic Structure

1.1.1 Currently working on

- lets players make moves and check when a betting round has finished

  TODO: think of all scenarios where different people are the last raiser (bigblind, small blind, no one, did some go all-in? etc...)
- next check for action to be valid
- the **Action** class should check if the action is valid -> see rules for raising

  test the isValidAction method for all scenarios
- create an **Information class** which holds all the information that is accessible to an AI.
- reseating players if needed, to balance game
- make a concept of what you need for your neural network and how the training is done.

  the training is purely done by evolving the weights and features there for no backprop algorithm or gradient calculation is needed

1.1.2 Ideas and Implementation Hierarchy

- A **game** is run simultaneously on **x tables**

  each **table** holds **y players**, a **deck of cards** and a **dealer (observer)**
- the **dealer** deals the cards, and holds informations about the game state
- the **statsKeeper** class keeps track of the opponent models
- Information class, that holds all the information of the tables but also for each player

Preferably each table runs on it's own thread, a coordinator makes sure that tables wait if reseating has to be done.

1.1.3 Open Bugs and Fixes

- is Raise action for > stack a valid allInAction? -> **for now it is**
- allInAction for less than a real raise should be treated like **call + extra**, meaning, the lastRaiseAmount stays the same but a new Raise has to be twice the lastRaiseAmount + extra
- if on flop two all in and two active players, and the first active player folds -> second active player shouldn't have the opportunity to doTurn
- change the way and all-in action is treaded (for now a raise of > stack size is not treated as all-in, it has to be exactly the same)

1.1.4 Things To Implement

- A UnitTest likle in oopoker_master which tests importent functions for correctness
- porting the neural network from python to c++ (using tensorflow api or write own neural network?)
- **Neural Network:** how to normalize input values if distribution is unknown???

  Answer:

  Additional feature (totalchips in game as BB), and other chip features also in bb Additionally normalize all chip features with the totalchips feature in bb and normalize other features to 0...1.
- Last layer is a softmax activation function so it gets a classification characteristic
- **Evolutionary Algorithm:** How to evolve our agents?
- how to order agents by places when multiple agents bust within one hand?

  Answer: guy with less chips has worst position...
- **Effective Hand Strength** as feature -> see how it is calculated and then try to do it with pokereval.h and pokereval2.h

  https://en.wikipedia.org/wiki/Poker_Effective_Hand_Strength_(EHS)_algorithm

  github for EHS calculation algo: https://github.com/Pip3r4o/Bluffasaurus
- interesting read on EHS for multiple opponents and general implementation strategies.. http://poker-ai.org/archive/www.pokerai.org/pf3/viewtopicfdcf.html?f=3&t=444&st=0&sk=t&sd=

1.2 Game Play Plan

Pregame

- get number of players
- calculate number of tables and cardDealers
- give players a buy-in and assign them to tables

Once all players have their buy-in and seat on a table, the game can start

Game Start

Preflop

- **Dealer:** assign dealer button to player
- **Players:** post SB and BB and antes
- **Dealer:** shuffle deck of cards
- **Dealer:** deal 2 cards to each player
- **Dealer:** tell first player after BB to play action

  Players: make a move

  Dealer: check if action is valid

  Dealer: repeat with next player
- **Dealer:** after all players acted:

  calculate statistics

  update table

Flop

- **Dealer:** burn one card, and deal flop
- **Dealer:** tell first player (SB) to play action

  Players: make a move

  Dealer: check if action is valid

  Dealer: repeat with next player
- **Dealer:** after all players acted:

  calculate statistics

  update table

1.3 Game Assets Classes

1.3.1 Information Class