Pokerbots Course Notes

Massachusetts Institute of Technology
6.176: Pokerbots Competition[*]

IAP 2020

Contents

[*]Contact: pokerbots@mit.edu

1 Lecture 1: Introduction to Pokerbots

This lecture is taught by David Amirault[1]. All code from lecture can be found at the public github.com repository mitpokerbots/reference-lecture-1. The slides from this lecture are available for download on Stellar.

At this lecture (and all others), we raffle off a pair of Beats Solo 3 Wireless Headphones. Attend lectures in person if you want a chance to win them!

We'd also like to thank our eight Pokerbots 2020 sponsors for making Pokerbots possible. You can find information about our sponsors in the syllabus and on our website, and drop your resume at pkr.bot/drop to network with them. Our sponsors will also be able to see your progress on the scrimmage server, giving you a chance to stand out over the course of our competition.

1.1 Class Overview & Logistical Details

There will be six 90 minute lectures running on MWF 1:00 – 2:30 pm from 1/6 to 1/17 in 54–100, and office hours will be held during the first three weeks of IAP[2]. There will also be a live scrimmage server for the first three weeks, on which you can challenge any other team in the class as well as our reference bots. Weekly tournaments will be held on the scrimmage server every Friday night, and there will be prizes for winning teams. The final tournament and event will be held on January 31, 2020, which is where the Pokerbots 2020 winners will be announced. The final event will also feature more prizes, an expert guest talk, winning strategy analysis, a chance to play against the bots, networking with sponsors and more! This year's Pokerbots prize pool is over $32,000, distributed over many different categories—the syllabus lists many of the categories we will be awarding. The six lecture topics will be as follows:

1. Introduction to Pokerbots
2. Poker Strategy + Bot Demo
3. Inference
4. Game Theory
5. Advanced Topics I
6. Advanced Topics II

To receive credit for the class, you must submit bots to the scrimmage server. Your bot for each week has to defeat your bot from the previous week, as well as a random bot in a one-shot tournament. At the end, you must also submit a 3-5 page long strategy report[3]. More guidelines will be announced later in the course[4]. Take special care to read the Rules and Code of Conduct on Stellar.

1.2 Introduction to Poker

For this section, we will be talking about the game known as heads-up no-limit Texas Hold'em ("poker"). The objective of poker is to win as many chips as possible. Players bet into a pot in several rounds, and the pot is won by the player with the better poker hand at the end. In a single betting round, the first player can either bet 0 (check) or any amount between the "big blind" and number of chips they have left. If they check, action passes to the second player, and if they bet, the second player can *fold* (quit the round), *call* (bet the same amount as the first player), or *raise* (bet more than the first player, up to the number of chips they have left). A player's final "poker hand" is determined as the best five-card hand that can be formed out of seven cards: their unique two *hole* cards and five shared *community* cards.

[1]Email: davidja@mit.edu.

[2]The schedule and locations for our lectures and office hours can be found on Piazza in post @8.

[3]You are welcome to include images or code snippets in your final report, as well as discuss strategies you attempted that did not pan out. This is an open ended report, and is for us to gain insight about how you approached the Pokerbots challenge.

[4]All class details are included in the syllabus, available on Stellar at pkr.bot/stellar.

The first betting round is special because it begins with blinds, a forced amount players have to bet. The next time around, there is no minimum amount.

The structure of a game is as follows: the game begins with each player receiving two hole cards. The first betting round takes place, and then the "flop" (three community cards are revealed). After another betting round, there is the "turn" (a fourth community card is revealed). Another betting round occurs, and there is the river (a final fifth community card is revealed). The last betting round takes place next, followed by settlement (cards are revealed and the pot given to the winning player).

The possible poker hands are displayed in the slides, in order of best to worst hands starting with the top left hand. The final hand is called "high card," which is none of the displayed ones. Even within each hand, there are tiebreakers if both players have the same hand. Make sure to look up, using one of the provided resources, which hands are better when building your bot.[5]

1.2.1 2020 Variant: "Permutation Hold'em"

This year, our variant of poker is Permutation Hold'em. Permutation Hold'em is based on the popular poker variant Texas Hold'em with a modification that the card values are randomly permuted each game. A poker hand's strength is determined using the permuted card values rather than the card values seen by the players. A random permutation of the card values 2–A is sampled at the start of each game, and the permutation is fixed for all rounds of the game. The sampling process uses a fixed prior distribution. The permutation may be different from one game to the next. Details of the prior distribution used to choose permutations can be found in the variant write-up on the 2020 class Stellar site.

Betting is also different from Texas Hold'em in that players have the same bankroll during every round (200 for this variant). Winners are calculated in terms of their change in bankroll aggregated across rounds.

1.3 Skeleton Bot Setup

1.3.1 Github and Version Control

GitHub is a version control/code management system. Using it, *clone* the public Pokerbots repository mitpokerbots/engine-2020 (available on github.com) to get started. If you've successfully set up Git on your machine, this can be done by navigating to the directory where you'd like to keep the code and running the command

```
$ git clone https://github.com/mitpokerbots/engine-2020.git
```

Skeleton bots for all supported languages are included in this repo.

We recommend that you create a new private repository of your own to code in on github.mit.edu, and then set this up by cloning it into your working directory and copy-pasting the engine files (`engine.py`, `config.py`, `cpp_skeleton`, `python_skeleton`, and `java_skeleton`) into the clone of your own repository. On the Pokerbots GitHub, the folder "python-skeleton-2020" contains the Python 3.7 skeleton bot. There are also Java and C++ skeletons available on our GitHub repository.

To upload code from your machine, you have to create a *commit*. To make a commit, *add* the changed files you want to push, describe it with a commit message, create the *commit*, and *push* the commit online. Your partners can now *pull* your changes to their own desktops. You can learn more about this workflow at http://web.mit.edu/6.031/www/fa19/getting-started/#git. For example, after editing `player.py`, you would push it to Github with the following commands

```
$ git add player.py
$ git commit -m ``made our bot super cool''
$ git push
```

Table 1 lists some common Git commands for your convenience.

[5]The following resource is great for learning more about Texas Hold'em: pkr.bot/poker-rules.

Command	Description
git clone **your_link**	Downloads code from remote
git status	Print the current status of your repo
git pull	Pulls latest changes
git add **your_files**	Stages changes for commit
git commit -m "**your_message**"	Commits added changes
git push	Pushes your changes to remote

Table 1: Important Git Commands

1.3.2 Connecting to the Scrimmage Server

To use the scrimmage server, go to pkr.bot/scrimmage. There, you can create or join a team with your one to three partners. To upload a bot, go to the "Manage Team" tab. Bots must be submitted as a zipped file, which you can easily do by going to "Clone or download" on your online GitHub repository and downloading your repo as a zipped folder. After you set one of your uploaded bots as your main bot, you can challenge any of the teams on the scrimmage server. If a team has a higher ELO rating than you, your challenge will be automatically accepted—otherwise, they must accept your challenge request.

1.4 Testing Your Bot Locally

To test your bot locally (without using the scrimmage server), you have to download the engine—again using GitHub. The engine consists of two files: `engine.py`, which runs your bot, and `config.py`, which contains parameters for your bot. The default parameters for the game, `BIG_BLIND` and `STARTING_STACK`, are the values we will be using, so don't change those. You should feel free to change `NUM_ROUNDS` and the time-related parameters; however, `STARTING_GAME_CLOCK` is capped at 30 for our tournament, since we do not want bots pondering for a long time. Before running the engine, you must specify which bots you wish to use in your local game. This is done by providing the file path of each bot in `PLAYER_1_PATH` and `PLAYER_2_PATH` (you may use the same file path to pit a bot against itself). To run the engine, we will be using command line. Change the working directory as needed to your engine folder, and then run `python3 engine.py` [6]. You will be greeted by the MIT Pokerbots logo, and your game log(s) will be output.

Looking at the game logs, we can see every action taken in each game and the permuted cards with the corresponding true values. The log shows the action chosen by each bot. You can also see that the log notes the flop, turn and river. The engine does not tell you what type of poker hand each player has—this has to be determined by yourself—but it will do the calculations and tell you who won.

In addition to the game logs, you will get a dump file, and this will be done for each bot. If you put any print statements in your bot, they will show up in the dump file. If you have an error in your code, the error descriptions will be in your dump file as well.

1.5 Overview of Skeleton Bot Architecture

Now, we'll look at the Python skeleton bot itself (`player.py`). The function `__init__` simply initializes the player object when a new game starts, and is useful for initializing variables that you want to access over the course of the game. The function `handle_new_round` is called at the start of every round, and the parameter `game_state` contains some information about the current state of your game. The function `handle_round_over` is called whenever a round ends, and is thus great for updating variables using the information from the last round played.

The `get_action` function determines your bot's move based on the presented information in the function's parameters—it's where the magic happens. Each of the commented-out lines contains important variables and their respective explanations, which you will likely find very useful as you develop your pokerbot.

[6] Depending on your setup, the command used may vary. Please refer to the setup Piazza post.

1.6 Coding Lecture 1 Reference Bot

We will now be implementing a basic inference bot and submitting it to the scrimmage server. The first thing we need is a way to evaluate how good cards are—we will do so by keeping track of how often we or our opponent wins for each card rank we or they, respectively, hold using dictionaries. We initialize two dictionaries in our `__init__` function: a `self.wins_dict` and `self.showdowns_dict`. Note that we map each possible card value to a 1 in the first dictionary and a 2 in the second dictionary; this is to avoid possible divide by 0 errors, as we care about the ratio between the two dictionaries for a card's win frequency. In our `handle_round_over` function, we will take care of modifying these dictionaries using the information from each round played. We first note our bankroll change from the previous round in `my_delta`, as this will tell us who won the round (positive: us, negative: opponent); this logic will help us determine the winning pair of cards for our dictionary modification. We next want to make sure that we only consider cases in which there was a showdown between our cards and our opponents', as we need complete information to ensure that a pair of cards is actually better than another. The remaining code, available in the Lecture 1 Reference Bot code base, modifies the two dictionaries using the delta and showdowns-only condition; each time, we must update the appearance of all four cards by one, and the winning frequency of only the two winning cards by 1 as well.

Finally, we look to the `get_action` function, where we will actually implement our pokerbot's actions. For the Lecture 1 Reference Bot, the only logic we implement is raising (whenever allowed) by the minimum amount when both of our two cards have a win rate above $\frac{1}{2}$. We calculate the winrate of our cards by taking the ratio of their respective values in the two dictionaries we have been updating throughout the game. If we cannot raise, we simply check-call.

We run the engine using this bot against the random bot and analyze the results. This time, we win against the random bot by a much greater margin than before; this is an example of how even basic strategy can help significantly.[7]

[7]Note that this strategy is deterministic, which is actually undesirable—in the next class, we will cover why. If you are playing purely based on how good your hand is, your opponent will be able to tell and then dominate you.