IT3105 Project I: Texas Hold 'Em

1 Overview

Your assignment is to design and implement a poker-playing simulator for the (very popular) game of Texas Hold 'Em. A complete description of the game can be found at any number of web sites, including www.pokerlistings.com, www.learn-texas-holdem.com, and Wikipedia.

You are free to use any computer language for this assignment. Some supporting Python code (for computing poker-hand power ratings) is provided, if you choose to use Python or want to use our code as pseudocode for translation to a different language.

The project can be done alone or in groups of 2 students (but no more than 2).

There are 3 phases to the project, worth 25, 45 and 25 points, respectively (with the final 5 points based on the general quality of the report as a whole).

Phase I	The basic simulator for k-player Texas Hold 'Em. **25 points**.
Phase II	A *pre-flop rollout simulator* used to generate the winning potential of all hole-card pairs, and a procedure for calculating the hand-strength of the combination of hole cards and community cards. **45 points**.
Phase III	An *opponent modeler* that a) records associations between game situations, opponent actions and the strengths of their hands, and b) uses those associations to estimate an opponent's hand strength in the current game. **25 points**.

Please read the lecture notes entitled *ai-poker-players.pdf* very carefully in order to fully understand the concepts discussed in this project description.

2 Phase I: The Basic Texas Hold 'Em Simulator

Your simulator should accept any number of players, from 2 up to 10. All can be computer players, although you are free to include options for one or a few human players.

Each player should begin with a sum of money and be allowed to play as long as desired. It is fine if players *go into debt* by having negative money; they can continue to play and place the same size bets as more successful players.

During each betting round, each remaining player (i.e., one who has not folded) should assess the power of their hand. This and any other information should be used to decide upon the current betting decision: fold, call or raise. In phase I of the project, this decision can be trivial, but it must at least be based on the hand's power rating. Prior to the flop, the power rating has little meaning, so any criteria whatsoever (even the flip of a coin) can be used to determine betting behavior.

Though concepts such as *left of the dealer* may not make much sense in simulation, you should have some concept of playing order, and it should be varied with each hand. You should either a) keep a fixed ordering of players and simply alternate among who bets first in each hand, or b) randomly shuffle the player ordering after each hand.

There is no need to designate a simulated player as *dealer*. Rather, you will probably want to have a main poker-manager object that handles all dealing and shuffling activities. A newly shuffled 52-card deck should be used for each hand.

The concept of *showing cards* during a showdown is also irrelevant in Phase I. The poker manager can simply compare the *power ratings* (see ai-poker-players.pdf) and distribute the pot to the winner(s). Only in Phase III will the revealing of cards during a showdown have significance.

The state of the table in terms of the shared cards, all hole cards, and the pot, should be displayed prior to each betting round. Then, during the betting, each player action should be displayed, as should the round's current bet. A fancy graphic interface is unnecessary, but all of this important game information should be printed in the command window. This information should be displayed during single runs of the system (during the demo and for debugging purposes), but it can be turned off when doing hundreds (or thousands) of runs to generate various statistics.

In phase I, time should be no problem: your simulator will probably be able to play thousands of hands in a few seconds. The computational demands will increase in stages II and III.

Warning: If your function for computing power ratings uses any list-sorting procedures that destructively modify card lists, then be sure to make copies of cards, particularly the shared flop, turn and river cards, when computing power ratings. Otherwise, you can run into all sorts of problems when using many of the common programming languages.

3 Phase II: The Pre-Flop Roll-Out Simulator and Hand-Strength Calculators

3.1 Pre-Flop Rollouts

As described in detail in the document *ai-poker-players.pdf*, a rollout simulator is a very effective method for assessing the winning probability of a poker hand. In this phase of the project, you will implement the pre-flop rollout simulator. You will then run it for the 169 equivalence classes of hole cards, and for all table sizes from 2 to 10 players. The number of rollout rounds per evaluation should be 100 (at the very least). Since this computation will be done off-line, you should be able to achieve 1000 or even 10000 rollouts in the course of a few hours.

The results from all these rollouts should be stored in a data structure similar to that shown in *ai-poker-players.pdf*, although you need not implement the exact structure shown. What is essential is that you have **one entry per hole-card equivalence class**, and that entry should house winning probabilities for each of the table sizes (2-10). You should not have one entry per hole-card pair, but per equivalence class. Given any pair of hole cards, your system should then compute the proper equivalence class and then use that class as an index into your data structure.

The pre-flop rollouts should only be computed once, during a lengthy off-line run. The contents of your data structures should then be dumped to a file such that they can be reloaded and used during any run of your Texas Hold 'Em simulator.

3.2 Hand-Strength Calculations

This is a relatively simple part of this project phase. It performs the basic hand-strength computations described in *ai-poker-players.pdf.* It should accept a pair of hole cards (H*) and 3, 4 or 5 shared cards. From these, it must analyze all remaining possible hole-card pairs and determine how often H* wins, loses and ties against those pairs.

3.3 Using winning probabilities and hand strengths

For phase II of the project, the fold/call/bet decisions of your players must be based on the information in the preflop-rollout tables or the calculated hand-strength, depending upon the stage of a hand.

As part of this phase, you must include players that use this phase-II information and those that do not. For example, you can have a table with 3 players using phase-II information and 3 players using a phase-I strategy. You must allow these strategies to play against each other for many hands (i.e. hundreds or thousands) and keep track of the winnings. Document whether the phase-II strategy(ies) had any advantage of the phase I approach(es).

4 Phase III: The Opponent Modeler

Up to this point, the system has essentially ignored any specific information about the opponents. Betting decisions are based purely on one's own cards and their potential strength compared to **possible** hands that opponents **may** hold. Of course, a player will never know her opponent's cards until an eventual showdown, but that showdown information can be used to influence betting decisions in **future** hands.

In this phase of the project, your system will accumulate statistics about player behavior. As discussed in detail in *ai-poker-player.pdf*, It will log associations between game contexts, player actions, and the hand strengths of players (as revealed during showdowns).

Since these associations will be entirely based on public information: game states, player actions (i.e., fold, call, bet/raise) and hole cards revealed in showdowns, it is sufficient to create one large data structure that comprises the model of ALL players. Each player can then have access to the models of all other players.

To showcase the opponent modeler, you will need to run your system through many hands, probably hundreds or thousands, in order to compile useful statistics on the different players. After that, you can turn off statistic-gathering and simply use the models for estimated-hand-strength information.

One way to differentiate between player strategies in this phase is to give some players access to the models while denying it to others. If your models house anything useful, then the players that use them should win more money in the long run. As discussed in *ai-poker-player.pdf*, the key output from the model is the estimated hand strength of an opponent; different players may use those values differently. A conservative player might fold if **any** of the estimated hand strengths of the remaining players are greater than her own hand strength, while a liberal player might raise if her hand strength exceeds the estimates of, say, half the other hand strengths.

Poker strategies, and poker psychology, are extremely complex subjects which you can only scratch the surface of in this project. However, to get full credit for this portion of the project, you will need to write code to create the player models and then show that the player models are being used to influence betting decisions in at least some of your poker-bots.

You must include at least one run (consisting of hundreds or thousands of games, depending upon your system's computational demands) in which strategies from phases I, II and III compete. Document whether the use of opponent models led to improved play compared to the phase I and II strategies.

In general, there is considerable room for creativity in this third phase of the project. A key design decision will involve the factors used in defining a *context* for the user models. Too narrow a context will produce too few hand-strength values, and too wide a context will capture so many diverse hand-strength values that their average will become insignificant and/or misleading.

5 Deliverables

Your entire system will be demonstrated at the end of this project period. For each portion of the project for which you intend to get credit, you must show a functioning program. In each phase, you should have at least 2 different player strategies that are used against each other. For example, in a 5-person game, you might have two players that bluff a lot (i.e. they like to raise on weak hands) and 3 who are conservative and only raise when they have good cards. The strategies will vary from phase to phase, since different types of information will be available in each phase, but each phase must have at least 2 different strategies.

A report of approximately 10 pages must accompany the project. The report should give a **high level** description of the system, with special focus on:

1. The basic structure of your code for each project phase,
2. The logic behind the betting decisions made by the players in **each** of the project phases that you complete,
3. The opponent models used in phase III, in particular, the contexts used to differentiate game situations,
4. The results of multi-hand (1000) runs for **each** of the project phases that you complete. These should **not** be screen dumps, but simple tables indicating each players total winnings. Five simple tables per project phase is all that is required (and desired) here.
5. A brief discussion of why (you believe) certain strategies were more (or less) effective than others for **each** phase of the project.

The report should not include code, but all of your source code must be uploaded to It's Learning in parallel with the report.

5.1 Point Breakdown

Your combination of demonstration and report will be worth a total of 100 points, which will be assessed in the following manner:

1. A well-functioning Phase I player that obeys all fundamental poker rules. **(20 points)**

2. Five different 1000-hand runs of a game involving 4 or more Phase-I players, with the results of each 1000-game segment summarized in a table showing the final resource (e.g. money) of each player. **(5 points)**
3. Modules for computing, storing (to file) and loading (for use in a poker game) the pre-flop rollout probabilities for k = 2 to 10 player games. Each rollout for each card equivalence class should involve at least 1000 rounds of simulated play. **(20 points)**
4. Modules for estimating hand strength for a pair of hole cards plus 3, 4 or 5 community/shared cards. This will involve testing the given hand against hands made from all possible pairs of hole cards (using the remaining 47 to 45 cards), as described in ai-poker-players.pdf and diagramed in Figure 15 of that document. **(15 points)**
5. Five different 1000-hand runs of a game involving 4 or more Phase-II players, with the results of each 1000-game segment summarized in a table, as above. **(10 points)**
6. Modules for performing basic opponent modeling. **(15 points)**
7. Five different 1000-hand runs of 4 or more players, at least of 2 of which use Phase-III techniques, while the others may use only Phase-II methods. Include the standard summary table of each 1000-game segment. **(10 points)**
8. The general quality of the report, which must cover all of the 5 topics mentioned above. **(5 points)**.

The assessment of the first 7 of these 8 components will involve both the demonstration and the report. For example, the 7th point may take a bit of run-time, so the demonstration will involve a small (e.g. 15-hand) run just to prove that the system works, while the complete set of 5 1000-hand runs will be done prior to the demo and documented in the report.

During the demonstration, the instructor will both a) look at various parts of the source code, and b) expect to see each phase running (without repeated crashing).

If you are working with a partner, EACH person should be capable of answering questions about any part of the code. These will normally be relatively general questions, so it's expected that both group members understand the basic functionality of all key components of the system. Failure to satisfactorily answer these questions can result in a loss of points for the project. Group members will receive the same grade for the project.