Automated Heads-Up Poker Player

Hossein Karkeh Abadi Reza Takapoui

October 21, 2014

1 Motivation

In this project, we aim to discuss implementing an automated heads-up poker player. Heads-up poker is a form of Texas Hold'em poker that is played between two players. At the beginning of the game, each of the two players are dealt two cards and the players take actions according to their cards. As the game goes on, some other cards from the deck are revealed and the players take actions accordingly. Eventually, the player who can make better combinations of the cards on the table with their cards wins the hand. For more information please refer to the Wikipedia page on poker.

We will break the problem into a hierarchy of complexity levels. In the simplest form, we will work on the case in which we can only take passive actions (only call or fold, as opposed to bet aggressively), assuming that the other player takes actions which are randomly drawn from a distribution with given parameters (that depend on their card and the revealed cards.) In the next steps, we will work on the trade-off between exploration and exploitation to learn the model parameters when we do not know the actual distribution for the other player's actions. If time allows, we will move on to the case where we can bet aggressively too.

2 Model

Let C denote the set of cards. (Each card can be represented by a number and a suit.) Let $\mathcal{X} : \Omega \to C \times C$ be a random variable representing the opponent's cards in a particular hand. At any step i, the opponent takes an action A_i which is a realization of a random variable coming from a distribution which depends on her cards X, the set of revealed cards O_i, and the history of the hand H_i (previous actions) so far. For example, in the limit poker $A_i = \{\text{no bet}, \text{bet \$1}\}$, whereas in unlimited poker, $A_i = [0, \text{the opponenet}'\text{s stack}]$. Then when it's our turn to act, we take an action based on the observations A_i, O_i, and H_i. (In the simplest case, this action can only be calling or folding.) The dynamics of the game only allows us to see the opponent's cards X, if we call her bets in all steps. So, there is an implicit trade-off between exploring the distribution of our opponent's actions by calling her bets, and exploiting our acquired knowledge about this distribution.

We can evaluate the performance of our model by the regret function, which quantifies the gap between our collected rewards (winnings) and the best possible rewards. In the next steps, we will consider the cases where we can bet aggressively. We can also consider a broader family of distributions that can incorporate the dependency of our previous actions as well. Ultimately, we can evaluate our models for the more complicated case, by playing against humans or other automated poker players.

3 Approach

A baseline for our model can be ϵ-explore policy. An oracle can be the case where we can see the opponent's hands. The team members believe that our knowledge from CS221 especially reinforcement learning, Q-learning, SARSA, exploration-exploitation, Monte Carlo and function approximation can be extremely helpful in this problem. We also think that it is a difficult task to build up the hierarchy of models from simple ones to more complicated ones and model the human behavior.

We will mostly use Python for our simulations. There are a family of modules in Python that provide an interface to hand calculators and game simulators. There are plenty of automated poker players and our goal is to compare the performance of different reinforcement learning methods with the state of the art algorithms.