Reinforcement Learning for Texas Holdem Against Non-Adaptive Opponent

Yuan Wang(yuangw866@stanford.edu) & Kaiyu Zhao(kaiyuz@stanford.edu)

URL for Presentation: https://youtu.be/3K6yU6k7POc

Introduction

- Texas Holdem is a challenging game that involves both imperfect information and adversarial agents. In this project, we trained a reinforcement learning agent ("RL agent") to play a reasonably good game of poker against random, baseline and oracle players.
- Given the complexity of the real game of poker, we simplified the game based on the following rules:
  - Only two players in the game.
  - Each player can only bet / raise in fixed increments.
  - The deck is perfectly shuffled every round.

Challenges

- Creating the game engine from scratch given the speciality of our objectives.
- Adapting the Reinforcement Learning framework to create a simulation environment for a two player adversarial game.
- Poker is a complex game with a large state space (even in this simplified form), so we need to design a Q-estimator / feature extractor that allows the RL agent to estimate the Q value of (state, action) pairs not seen in training.

- Simplifying the game of poker while retaining some interesting challenges for a RL agent. For example, we would evaluate the utility after each round.

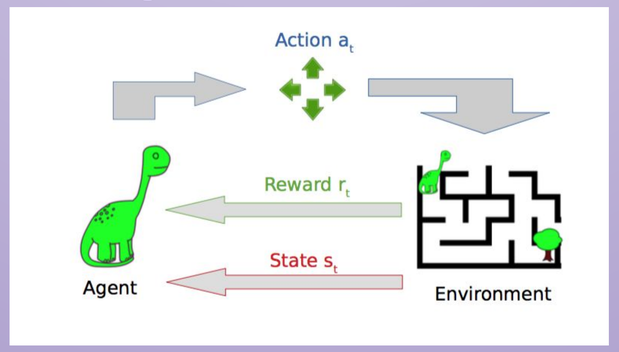
Approaches

Fixed-Policy Agents

- Apart from the game simulator and the environment for the 2 player reinforcement learning, we created the following fixed -policy agents.
  - **Random Player**: takes one of the legal actions randomly.
  - **Baseline Player:** evaluates the strength of its own hand and community cards and takes an action based on this evaluation.
  - **Oracle Player**: has perfect information about each player's hand and the 5 community cards (including undealt cards). The Oracle Player will fold if it is going to lose and takes the most aggressive action if it is going to win.

Reinforcement Learning Agents

- Another import part is that we created a RL agent with a linear-regression Q estimator. The underlying algorithm is the following equations based on Q-Learning Algorithm:
  - On each (state, action, reward, newState)
    - $Q_{opt}\hat{}(s, a) \leftarrow (1-\eta)Q_{opt}\hat{}(s, a) + \eta(r + \gamma V_{opt}\hat{}(s'))$
    - $V_{opt}\hat{}(s') = \max_{a' \in Actions}(s')\, Q_{opt}\hat{}(s', a')$
- The basic diagram for RL:



Results & Analysis

- **Between Fixed-Policy Agents**

Senario	Average Utility
Baseline V.S. Random	0.19
Oracle V.S. Random	0.79

Clearly, among these fixed-policy agents, the performance are ordered are oralce > baseline > random.

- **RL Agents V.S. Fixed-Policy Agents**

Senario	Average Utility
RL V.S. Random	0.35
RL V.S. Baseline	0.12

The performance for RL agent works better than random and baseline agen. This proves the efficiency of the RL!

- **Various Exploration Probability for RL V.S. Random**

Exploration Probability	Average Utility
0.1	0.23
0.2	0.18
0.3	0.09
0.4	0.10

This expl prob decides the route that the RL agent plays to avoid not stucking in local optimal. Our investigation shows the value in (0.1, 0.2) is most reasonable.