COMP423 - Reinforcement Learning and Dynamic Optimization
Poker Project Part 1 Report

Pantourakis Michail AM 2015030185
School of Electrical and Computer Engineering
Technical University of Crete

Date submitted: 30 June 2023

1 Introduction

A significant step towards understanding today's advancements in Reinforcement Learning (RL) is by first understanding the fundamental theory of Markov Decision Processes for modelling decision making problems and the "exact" model-based algorithms that optimally solve these problems. However, these approaches are limited by their requirement for complete knowledge of the environment (model). A major breakthrough in RL was thus the development of the Q Learning algorithm, the first model-free learning algorithm with guaranteed convergence to the optimal policy.

In this report I present my course project implementing these algorithms and analyzing their results on a simplified, "toy" version of poker. I will start by first establishing the rules of this simplified game, the opponents designed as part of the environment, my model representation and how I implemented all of them in Python. With the problem formulation being established, I will then continue with how I applied Policy Iteration (model-based) and Q Learning (model-free) algorithms to this problem, reporting key observations, and most importantly, analyzing their behavior due to the nature of this game and what theory predicts.

2 Environment

2.1 Simplified game rules

Both "exact" algorithms are limited in solving problems with a relatively small number of states and actions. As poker is characterized by numerous parameters (number of players, position/order of players, legal actions, bidding round number, betting chip/token allowance, hidden and public cards, winning card combinations), the state-space quickly has a large number of states. To alleviate this issue for the purposes of this project, a simplified version of the most popular poker variant, Texas hold'em was used. Since Texas hold'em is a well-known game and its rules are also summarized in the project description, I will only provide a quick summary of its rules here, with emphasis on the unique deviations assumed here.

First of all, a dumped-down version of heads-up limit Texas hold'em is used as a basis, thus only 2 players participate. Both players start with a card in hand (not the normal of 2 cards) and with betting the same mandatory amount of 0.5 tokens (namely no small/big "blind" difference).

The game continues with a bidding round. Since this is a limit variant, either 1 (actions "bet" or "raise") or 0 (actions "check" or "fold") tokens can be placed by a player per turn of action, with a maximum of 2 tokens placed in total by each player per round. If a player folds, the other player is an instant winner and is rewarded with the full amount of tokens bet so far. Otherwise, 2 cards are further drawn and are publicly revealed, and a second bidding round, similar to the first one, begins. Unlike the real game, the order (position) of players in which they are required to take action remains the same for both rounds. For simplicity, no further rounds and card draws take place. Table **??** summarizes the legal order of actions in all game states.

Assuming no player has folded until the end of round two, the winner is then decided by comparing their "hand strength". To reduce the number of possible combinations, only cards with rank 'T', 'J', 'Q', 'K', 'A' (in ascending order) are available in deck (20 cards in total, 4 suits per rank). This simplification leaves only three winning combinations of hand and public cards: 1) 3 cards of the same kind (rank), 2) a pair of same kind, 3) no pairs. The rank of the pair or hand is used as a tie-breaker in rules (2) and (3). If all tie-breakers fail, the result is a tie and the tokens placed by both players are returned to them.

+++ Add table

2.2 Opponents as part of the Environment

Two different types of "static" (non-adversarial) agents were created in this project. These opponents serve as both necessary components of the full state-space formulation required by Policy Iteration, and benchmarks for the performance of Q Learning.

The first opponent, hereafter called *Random agent*, is a completely randomized agent. Each time an action is required by it, it randomly picks -with equal probability- one of the actions allowed according to the game's current phase. Since the Random agent disregards any other card and opponent information, its opponent cannot infer any meaningful information from its actions.

The second opponent, named *Threshold agent*, is a completely deterministic agent, using only the strengths of its cards to determine its next action. Table **??** summarizes the action this agent will perform at any possible step. In contrast to the Random agent, each and every action of the Threshold agent provides information on the range of possible ranks held in its hand. Therefore, its predictable nature can be used against it by any agent trained against it.

+++ Add table

- State-action representation
- what was adapted from rlcard

3 Model-based solution: Policy Iteration algorithm

- State space creation
- Implementation
- Average Results
- Analysis of Optimal Policy per opponent
- Demonstration of optimality

4 Model-free solution: Q Learning algorithm

- Implementation
- Analysis of hyperparameter tuning
- Average Results
- Convergence analysis

5 Conclusion

- Main point for game size & performance of algorithms
- Suggested next steps/improvements