Imperfect Information Games: Modeling Responses to Irrational Behaviour using Deep Learning

Anonymous CVPR submission

Paper ID ****

Abstract

In this project we consider a complex, imperfect information, zero-sum game of Leduc poker between two players who have differing computational abilities. One player is able to approximately compute the optimal Nash Equilibrium strategy, while the other is forced to resort to computationally simple strategies. We will quantify the advantages that the optimal strategy has over common, simple strategies that have been utilized by people historically and discuss which of these simple strategies fares best. Additionally, we will ask what informational advantages would bring the most benefit to the computationally simple player. Concretely, we consider the implementation of DeepStack against several common poker strategies.

1. Introduction

Modeling strategies in imperfect-information games, such as Heads-Up No-Limits (HUNL) Poker and Leduc Hold'em, is challenging as the full state (the cards of the opponent) is unknown. Because of this challenge, imperfect-information games require more complex reasoning as each decision during play depends on a probability distribution of the opponent's private information, which can only be discerned through the opponent's action history.

Recent breakthrough work (see [6], [2]) has found success in optimizing towards Nash-equilibrium strategies, yet previous work has not exploited the average player's tendency to deviate from rationality in the context of games [5], exhibiting risk-aversion, risk-affinity and changes in risk-tolerance over the course of play [4]. The objective of this work is to identify any playing profiles in which semi-rational, irrational or historical training gives a player an advantage over the Nash Equilibrium strategy used by DeepStack, and more generally to evaluate the performance of several well-known player deviations from rationality against purely-rational play.

2. Related Work

Counterfactual regret minimization has been the foundational approach to improving performance in imperfect information games. Beyond this approach, two notable approaches have been explored to overcome the challenges of states lacking unique values.

2.1. Counterfactual Regret Minimization

Recursive reasoning approaches to solve imperfect-information games have been explored since 2008. Counterfactual regret minimization (CFR) has been the most notable technique that exploits the degree of incomplete information in an extensive game and uses self-play to recursively reason.

Self-play enables an AI agent to adapt its playing strategy against itself over many iterations and allows the AI agent to leverage "regrets" of prior playing decisions to inform future decisions. The AI agent's goal is to optimize its game tree traversal based on its opponent to converge on the Nash equilibrium strategy. The limitation of this approach comes when dealing with larger game trees where such computation during play is too expensive. In these cases, abstraction is used to simplify the game and then the simplified game is approximately solved via tabular CFR. Two types of abstractions have been previously employed – information abstraction, where information sets are bundled, and action abstraction, where a subset of actions are removed from game tree model.

Recent work has explored an innovation on the tabular CFR approach that introduces Deep CFR, a form of CFR that uses function approximation with deep neural networks to approximate the tabular CFR behavior without the need for game abstraction [1].

2.2. DeepStack

AI DeepStack used the technique of modifying the definition of state into a joint belief state across all players [6], which proved inferior to the top two HUNL Poker players. DeepStack's architecture consists of a standard feedforward neural network that has seven fully connected hidden layers, each with 500 nodes and parametric rectified linear units (32) as output. This neural network is then embedded in another network that imposes counterfactual regret minimization to satisfy the zero-sum property.

2.3. DeepStack-Leduc

Extending the approach of [2], Schmid et. al re-implemented the DeepStack AI agent for HUNL Leduc Hold'em. This adaptation of the work to a simplified game laid the groundwork for rapidly exploring the performance impact of introducing irrational and semi-rational playing profiles in a narrower gambling context.

3. Problem Formulation

3.1. Nash Equilibrium Strategy

The Nash Equilibrium strategy represents a profile of strategies in imperfect-information games in which no player can improve by shifting to a different strategy. Mathematically, a Nash Equilibrium in principle can be found by solving simultaneous equations:

$$x_1^* = \max_{x_i \in X_i} f_1(x_1^*, x_2)$$

$$x_2^* = \max_{x_i \in X_i} f_2(x_1, x_2^*)$$

Where x_i^* represents the optimal action Player i should take to converge on the Nash equilibrium of the game, x_i represents the action taken by Player i, X_i represents the set of all possible strategies Player i can take, and f_i is the function that maximizes the pay-off to Player i in response to the opponent's action.

When the opponent plays the Nash Equilibrium strategy the value of irrational, semi-rational playing profiles as well as psychological tactics during play (i.e. bluffing) may undermine the opponents ability to discern the true value of the players hand. This leads to suboptimal decisions made by the opponent during play as the opponent attempts to maximize the payoff based on the assumption that x_i, the action taken by the opponent is attempting to also converge on the Nash equilibrium. Irrational behaviors in play enables a player to influence the opponents actions by deliberating misleading the opponent.

3.2. Leduc Hold'em

To explore irrational, semi-irrational and psychologically manipulative AI playing agents, a simplified Poker game will be used known as Leduc Hold'em. When compared to HUNL, Leduc Hold'em presents a smaller game tree and enables faster Nash equilibrium computation, and reduces playing time and drastically reducing time required to learn.

Leduc Holdem is a two-player poker game that uses a deck of six cards consisting of two Jacks, two Queens and two Kings. The deck is shuffled prior to playing each hand and the game begins with a one-chip ante and a single private card dealt to each player. An initial betting round takes place followed by the flop, where a single public card is displayed from the deck. A second betting round transpires before the showdown where both players reveal their private cards. The player that has a pair with the public card wins the pot. In the case where neither does, the highest private card wins. In the case where both players have a pair, the players split the point.

3.3. Limitations of Rationality in Poker

In traditional poker, understanding the risk-tolerance of your opponent and their changes in risk aversion/affinity throughout the course of play is critical to winning. Successful, amateur players attribute a majority of their success to a refined ability to read their opponent far more than their ability to predict the potential value of any hand. Common strategies in poker include bluffing, evaluating risk, and trying to limit exposure to exploitation. These are all computationally-simple attempts at a similar resolving process to what DeepStack simulates.

How should a player behave when faced with a DeepStack opponent? A regular player, with no access to a computer and Lua installation, certainly cannot play the Nash equilibrium. Which common playing strategies perform well against DeepStack? In addition, what are the most

Player Profile	Type	Description
Rocks	Rules-based	Risk-averse player with strict folding logic.
Passive-Rocks	Rules-based	Risk-averse player with nuanced folding logic.
Mild Adaptive Rocks	Intelligent	Player that adjusts behavior based on success variance for state
Strong Adaptive Rocks	Intelligent	Player that adjusts behavior based on success variance for state
Randomized Bluffer	Random	Risk-tolerant player that randomly bluffs in play.
Semi Bluffer	Probabilistic	Risk-tolerant player that raises based on the probability of the opponent folding.
Aggressive Bluffer	Probabilistic	Player that liberally raises based on the probability of the opponent folding.

Table 1. List of the irrational playing profiles explored.

important things a player should try to figure out about an opponent (informational advantages)?

4. Methodology

Our approach will explore rules-based, probabilistic and intelligent, semi-rational AI agents. Each irrational and semi-rational agent will compete with a pre-trained DeepStack. There are four types of personalities in poker categorized based on their level of risk-tolerance (i.e. Loose vs. Tight) and their level of aggression (i.e. Passive vs. Aggressive). Within these broader definitions exist two playing profiles known as Rocks and Passive Rocks that will be modeled in this approach. Simultaneously, this work also explores three broadly defined bluffing behaviors in poker, differing on bluffing logic.

The relative risk-aversion of each of the playing profiles can be modeled in an adapted Domain-Specific Risk-Taking (DOSPERT) Scale.

4.1. Player Profiles Explored

Table 1 outlines an initial list of player profiles explored in this project, with a focus on modeling risk-aversion and bluffing behaviors that are typically found in traditional poker.

Rocks represents a very risk-averse (i.e. ”tight”) personality type that will fold in most cases when a high value card or potential pair does not exist. In the strategy space of Leduc Hold’em, this translates to playing only when the first card is a King. An example scenario of this player is found in Figure 1.

Passive Rocks will bet call or bet in some cases beyond the the highest value card. In the strategy space of Leduc Hold’em, this translates to playing hands that consist of a Queen or a King. Both of these traditional risk-averse playing profiles can be easily modeled using a rules-based approach.

Extending this rules-based approach, the Randomized Bluffer will simulate a coin toss to determine whether to bluff or logically.

4.2. Modeling the Bluffer Player Profiles

In order for the bluffing players to know when a raise is a ”bluff” (as opposed to just a rational play) the player must have some idea of rational play. Thus, we build the logic for bluffing on top of DeepStack. The randomly-bluffing player uses re-solving to determine the probabilities of each action. However, if ”raise” is a valid action, then, with probability .3, the player ignores the rationally-chosen action and does a large raise instead (and otherwise behaves rationally). The semi-bluffer, on the other hand, uses the probabilities of the opponent folding, based on the resolving of the game (the value estimates for the opponent when resolved). Instead of randomly bluffing with probability 0.3, it only irrationally raises if the estimated probability of the opponent folding is high.

4.3. Modeling the Semi-rational Player Profiles

The semi-rational player profiles will maximize a utility function that incorporates risk aversion on top of DeepStack as opposed to the standard approach which maximizes the risk-neutral expected payoff. The idea here is to incorporate traditional notions of risk-aversion from decision theory in economics and incorporate them into the evaluation of strategies. In order to do so instead of simply approximating the expected value of taking a given action in a given state we also will attempt to calculate the variance of outcomes and then vary the risk-aversion level and see how performance compares against the standard risk-neutral DeepStack.

4.4. Evaluation Criteria

To test performance, this work will evaluate the exploitability of each AI agents technique, the average backroll per hand and the average win rate per hands played. Exploitability of a strategy is defined as the measure of performance against a worst-case opponent. Exploitability is quantified as the difference in value of the game to a player and the opponents payoff/benefit if the opponent plays her best response.

5. Code/Preliminary Results

We expand upon two starter Github-repos, with code located at https://github.com/rawls238/deep_learning_project. The rational DeepStack player is cloned from the Leduc-DeepStack github [7]. The structure to create custom rule-based players in python, as well as evaluation criteria, is cloned from a final project from last year’s class which pitted DeepStack against Libratus [3]. We alter code at from DeepStack’s re-solver to create the players built on top of DeepStack, and write python code from scratch for the rule-based players. Instructions on how to play with these new players is included in the readme for our project. We so far have implemented 3 of the player profiles, which are listed in the readme along with some basic ones for testing purposes. The preliminary results of these matches (against DeepStack) are located in the outputs folder of the github repo.

Our next step is to figure out evaluation for these matches – what measures should we use for how successful one opponent is against another? We also plan to explore informational advantages and what effect these might have on playing against Deepstack – ie, how much information does a dumb player need to know in order to be able to BEAT DeepStack?

References

[1] G. Brown, Lerer and Sandholm. Deep counterfactual regret minimization. *eprint arXiv:1811.00164*, 2018. 1

[2] N. Brown, T. Sandholm, and B. Amos. Depth-limited solving for imperfect-information games. *arXiv preprint arXiv:1805.08195*, 2018. 1, 2

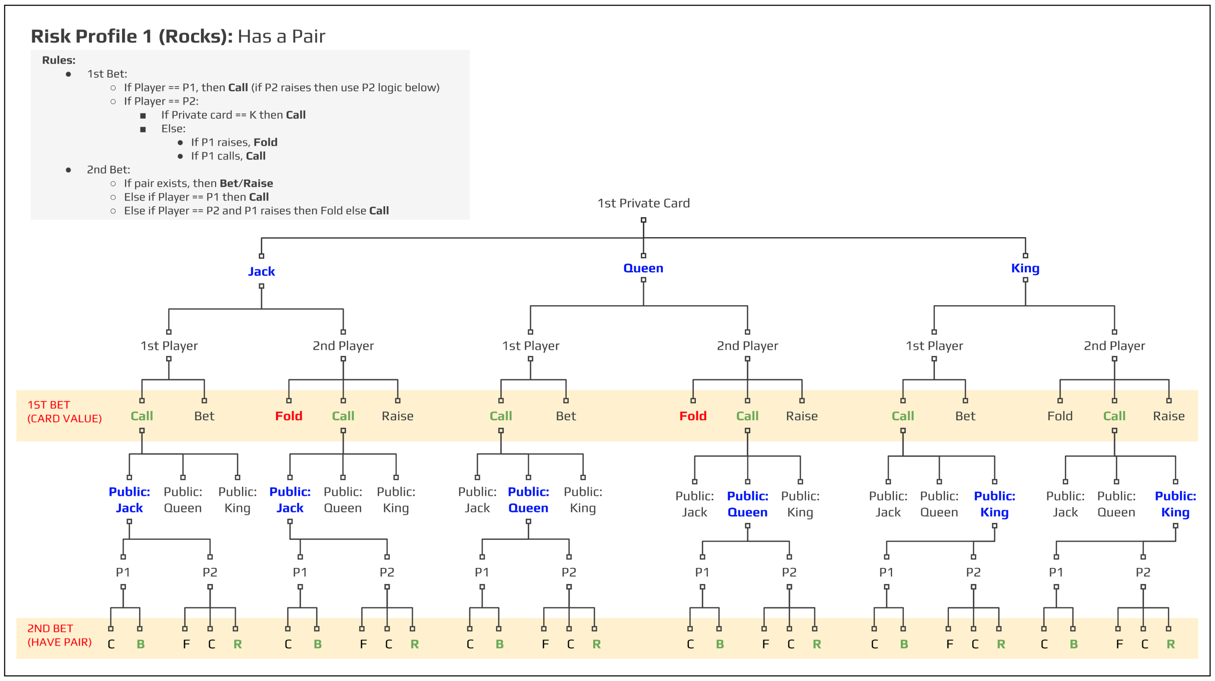
[3] Echlin, Porubin and Dunn. Coms 4995 Final Project - Fall 2017. https://github.com/mp3242/coms4995-finalproj, 2018. Online; accessed 25 October. 3

[4] D. Eil and J. W. Lien. Staying ahead and getting even: Risk attitudes of experienced poker players. *Games and Economic Behavior*, 87, 2014. 1

[5] J. K. Goeree and C. A. Holt. Ten little treasures of game theory and ten intuitive contradictions. *American Economic Review*, 91(5):1402–1422, 2001. 1

[6] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. 1

[7] M. Schmid. Deepstack for leduc hold-em. https://github.com/lifrordi/DeepStack-Leduc, 2017. 3



Risk Profile 1 (Rocks): Has a Pair

Rules:

- 1st Bet:
  - If Player == P1, then **Call** (if P2 raises then use P2 logic below)
  - If Player == P2:
    - If Private card == K then **Call**
    - Else:
      - If P1 raises, **Fold**
      - If P1 calls, **Call**
- 2nd Bet:
  - If pair exists, then **Bet**/**Raise**
  - Else if Player == P1 then **Call**
  - Else if Player == P2 and P1 raises then Fold else **Call**

Figure 1. Example of a Rocks Player Game tree in the scenario where a pair exists.