Texas Hold‘em Project Proposal

Ramirez  
CaMS  
Lewis University *Romeoville, United States*  
FidelRamirez@LewisU.edu

Lenz  
Consulting, Information Systems  
Lenz Family, LLC

*Homer Glen, United States*  
Thomas.Lenz96@gmail.com

*Abstract*—This document is a proposal to have the team develop AI software which simulates and autonamously plays a game of Texas Hold‘em.

Keywords—Poker, Texas Hold’em, Artifical Intelligence, Proposal, Opponent Modeling

# Introduction

Texas hold em’ is a variant of poker, a game of chance with limited information given at any time. Each player has two private cards in their hand, or their hole cards. In addition, the game can be split up into several sections: first, the game can be split up into its two most major stages: preflop and postflop. The “flop” is the first group of cards to be revealed in the community pool. The term “preflop” describes the stage of the game where the agent has the least information. In this stage, the only information available to the agent is the cards in its hand. These cards can be looked at by the agent at any time during the game. After the hole cards are dealt, there’s only one round of betting before the flop is revealed. Because of a limited amount of information, any bets made here tend to be tenative at best, and no success is guarenteed. Everything that occurs after the flop is added to the pool is part of the postflop stage. Since the flop is where the largest amount of objective information is gained, a heavy amount of pruning will take place. Specifically, at this stage, the agent can calculate its best and worst hands, assuming the unseen cards are optimal for each. This means the agent can ignore any path which involves it having a hand better than its optimal, and worse than its least optimal. The same cannot be done effectively for the opponent until the fourth card is shown, when all but three of the opponents cards are known. Before this, four cards are unknown, meaning the opponent can always match the best category of hands, a straight flush. However, the opponents least optimal hand can be found, and the tree can be pruned further. The possibility of analyzing betting habits will also become exponentially more effective postflop, should the agent be developed with the ability to analyze such data. Postflop, the game can be broken down further. After the flop is revealed, another round of betting occurs. Subsequently, the fourth card in the pool, known as “the turn”, is revealed, followed by another round of bets. Once all players are even, the river, or the final of the five cards, is dealt into the pool. The final round of betting finally occurs, followed by the showdown, where all the players’ hole cards are revealed, and the best hand is found for each. At each of these stages, save for the showdown, an agent has to make decisions based off of prior experiences or based off of what it has been taught previously in order to maximize its winnings while minimizing its losses during any hand. By applying mathematics we can decide when it is best to take any of our actions.

# Environment

## Visibility

The first thing we have to do is identify the environment and all the properties of it. The first property is that this is partially visible, since while the agent is not able to see what other agents have, it can see its own hand, as well as the community pool.

## Agency and Assumptions of Other Agents

The next property is that this is a multi-agent environment. Specifically, this is a competitive environment in a zero sum game as we are learning right now, and there are differences in how agents should behave according to each other. This leads us into considering game theory under the assumption that our opponent is rational as well.

## Determinism vs Stochasticism

Since we are working with probability, the environment that we are working in is going to stochastic. This means that even if we are rational and make no mistakes in our decision making based on statistics and probability we may still lose the goal though is to win more than we lose.

## Episodic vs Sequential

The environment is also episodic, since the agent should be making decisions based off of the current state of the board, its hand, and the pot. While an argument could be made that the amount of money invested into the pot should, under certain circumstances, affect what the agent does in response to a bet, it’s hard to justify explicitly programming in the lost-cost fallacy, and therefore, this will be avoided.

## Dynamicism

The environment can also be considered to be static. Because the rules of the game do not change, and the decisions made by the agent are not constrained by time, suggests a static environment. Since the performance score will not change with the time spent on making decisions, the environment is not semi-dynamic.

## Continious vs Discreet

As stated earlier, the game of Texas Hold’em can be broken into several distinct phases. These phases are made up of either a finite number of states, or children phases which this description would recursively apply to. Therefore, after the state tree is fully drawn out, there must be a finite number of states, making the environment discreet.

## Known vs Unknown

Texas Hold’em has a specific set of rules which it is played by. These rules are composed of the enumeration of phases within the game, events that occur during these phases, and a heirarchy which is used to judge who wins the hand. Each deck has a set number of cards, and each card has exactly one match to another card in another deck. Through the knowledge of the deck, its contents, the heirarchy of hands, and the community pool, the probability of which agent will win can be determined. Because the agent will have knowledge of all of this information, the environment is fully known.

# Foreseeable Obstacles

As a result of the environment, there are several issues that will needed to be worked around or solved. The first of which being that are working with limited information. The fact that we will not be able to fully see the game will mean that there will be a certain amount of uncertainty no matter what we do. However, that can be somewhat reduced by using statistics and making some assumptions. A primary example of this would be comparing risk versus reward. The agent can figure out the optimal and least-optimal hands each agent can play, calculate the probability of each, and figure out the probability any given agent wins. This alone removes a vast amount of uncertainty in many cases. However, this can be taken a step further. The agent can then use the probability of its victory, along with the amount of money at stake and the bet required to stay in the game, to figure out whether it’s smart to call, raise, or fold. Another way to make decisions is to perhaps somehow teach the agent. This can be done in a few different ways, but currently we are undecided how we would go about it and how we would implement it. There was a study done by that actually used both concepts mentioned previously, where they taught an AI how to make choices depending on how the game state is. In that study, they used previously recorded results of games in order to teach the AI. They were able to do this by clustering according to certain attributes that were recorded. After being trained long enough, for the right amount as well, when given a state, the AI was able to make choices based off its previous knowledge. The example given in the study, however; was only concerned with pre-flop choices, not for the whole hand[1]. That being said, the proof-of-concept could be further expanded. This is shown by Carnegie Mellon University’s Liberatus, an AI system which beat the best of the best, winning nearly $1.8 million over 120,000 hands of poker. This dramatic success lead experts to conclude that AI has surpassed humans’ ability in strategic reasoning with imperfect information[2]. Another example of this would be DeepStack. This example will be particularly helpful, as one of its creators, Murray Campbell from IBM, shared some of his team’s tricks that made their system so successful. One of these tips is to only calculate a few steps ahead. While some systems will calculate as far as possible, and then use a best-guess algorithm to make its guess, this method will often lead to grouping together strategies that don’t work. By only worring about what occurs within a limited scope, these ineffective strategies are weeded out. On top of this, DeepStack uses neural networks and machine learning to continue to improve itself[3]. This leads us to another problem: actually learning how to implement an AI which can learn from information that we would already have somehow. Thankfully, there are sources that are available to help us learn in depth of how such an AI would learn and how to teach it, one source being the book that is assigned to this course.

# Final Words

To conclude, after examining the environment that we could be working in, this should be a rather challenging project to undertake. However, this is nothing that has not been done already, and there are more sources that exist covering the same topic and all of those can still be referenced if need be. In addition, most of what this seems to be is statistics, probability, and data mining, none of which is too difficult to learn on our own. While we may not cover it in class, we should be confident enough to tackle the problem. If we do find issues relatively early on the project, though, we may choose to somehow alter the environment. One suggested bit of feedback that we already recieved was to perhaps make the environment fully visible, and let the primary agent see what the opponent has in their hand. This would be a huge alteration to the project and would remove a significant amount of uncertainty in the game, if not removing uncertainty in its entirety. This will give more consistent results for a proof-of-concept. There may be other options or routes that we could do in order to perhaps make the project more manageable as well, but for now, we're going to be working in the given environment and attempt to make an AI which would have learned from prior games, based on the methods described previously in this paper.

# References

1. Teofilo, Luis Filipe, and Luis Paulo Reis. Identifying Player’s Strategies in No Limit Texas Hold’em Poker through the Analysis of Individual Moves . Portuguese Conference on Artificial Intelligence, 7 October, 2011, arxiv.org/pdf/1301.5943.pdf.
2. Hsu, Jeremy. “AI Decisively Defeats Human Poker Players.” IEEE Spectrum: Technology, Engineering, and Science News, IEEE, 31 Jan. 2017, spectrum.ieee.org/automaton/robotics/artificial-intelligence/ai-learns-from-mistakes-to-defeat-human-poker-players.
3. Riley, Tonya. “Artificial intelligence goes deep to beat humans at poker.” *Science | AAAS*, American Association for the Advancement of Science, 3 Mar. 2017, www.sciencemag.org/news/2017/03/artificial-intelligence-goes-deep-beat-humans-poker.