Matt Morgis

Peter Chapman

Mitchell McCann

Temple University

Dr. Pei Wang

CIS 3203 – Intro to Artificial Intelligence

December 2014

Artificial Intelligence Strategies Applied to Texas Hold’Em Poker

**Introduction**

Poker is currently the most commonly played card game in the world. There are many factors that contribute to its success, but the biggest contributor is the game’s raw fundamentals. There are many unknown elements forcing the opponents to observe each other and make good decisions, while at the same time adding a complex statistical variation to the game that gives even sub-optimal players a chance.

Our overall goal was to build a Texas Hold ’Em simulator that would give near perfect advice to a user using only the information available to them.

The basic properties that are included in a game of poker, such as dealing with hidden information, complex random events, and the fact that many human players are able to out-perform many machine implementations make this a very relevant topic in the field of Artificial Intelligence. Achieving success for such a non-trivial game would be a large step in the right direction for the field of A.I. in general, and has been attempted many times in the past decade.

**Overview of the game**

The game can be played between two and 10 players. A normal French, 52-card deck is used and up to 7 decks can be used to add complexity. Each player starts off with a set amount of money or poker chips. The game is made up of many handsthat are played. There are multiple rounds of betting during each hand. The game is over when one player has successfully acquired all of the money from each opponent.

At the start of each hand, each player is dealt two cards facedown. This stage of the hand is known as the *pre-flop* and the initial cards are called *hole cards*. A complete hand also involves the *flop,* which is three cards dealt face up, the *turn*, which is an additional card dealt face up, and the *river*, which is the final card dealt face up. Each player is to make the best five-card poker hand using the community cards dealt during the flop, turn, and river, and any number of their hole cards. There is a round of betting between each phase when cards are dealt.

There are four different betting options a player has at each phase*: fold, check/call, raise/bet* and *all-in.* If a player *folds*, the player must sit out of the rest of the hand and forfeit the money they already put in to the pot. If a player *checks or call*s, the player bets the minimum amount possible to remain in the hand. Alternatively, a *raise* or *bet* increases the current bet and forces other players to reply to stay in the hand. Finally, an *all-in* is when a player places all of their remaining money into the pot.

**Complexity of the game**

Complexity seems to arrive at every angle of the game and in every form. As each player and deck is added to the game, the game state drastically increases. There are 52 unique cards, and each deck increases that number. Players add complexity to the betting rounds, have an effect on the order the cards get dealt, and add more intangible factors like deception.

There are two types of Texas Hold ’Em that also play a role. The most common played version in casinos across the United States is *No Limit.* No-Limit allows players to go all-in, has no maximum bets and has an estimated game state of 1e^71 different game states. Meanwhile *limit* poker has betting constraints that bring the state spaced down to about 1e^18.

Poker is also a *nondeterministic* game, meaning there are many possible outcomes given the same broad game state. For instance, if 10 hands has the same five community cards, each hand would have a drastically unique outcome. However, in light of this, there are some outcomes that happen less likely (royal flush) while others (community high-card) are more common.

In order to reduce the complexity of the game and simplify our analysis we decided to use a limit game, keep it to two players and only use one deck. These constraints would help us determine strategies for each problem state throughout the each hand instead of a general solution for the entire game. We felt this would help us stay flexible and allow our advice to adapt as the game progressed.

**Initial Research**

Our initial strategy was dependent on two main sources of information. The first, an open-source poker hand evaluator that followed the *2 + 2 algorithm,* which was first published in a 1998 research paper titled “Opponent Modeling in Poker.” The algorithm is broken down in more detail below.

The second source of information was a table returning an evaluation of a player’s initial hole cards. According to many studies, the most important and biggest advantage a player can have is to know the value of your hole cards.

**Hand Evaluations**

To evaluate a player’s hole cards, a numerical value calculated from the possible 115 million hands using data mined methods is assigned to each card pair. We wrote a simple script that would return that numerical value given a two-card combination

Evaluating cards post-flop was done using the aforementioned open-sourced JavaScript poker evaluator. It took a three, five or seven cards as input and mapped it to a unique integer representing the score of the hand. A sample call and return is below.

PokerEvaluator.evalHand(["As", "Ks", "Qs", "Js", "Ts", "3c", "5h"]);

*//{ handType: 9,*

*// handRank: 10,*

*// value: 36874,*

*// handName: 'straight flush' }*

PokerEvaluator.evalHand(["As", "Ac", "Ad", "5d", "5s"]);

*//{ handType: 7,*

*// handRank: 148,*

*// value: 28820,*

*// handName: 'full house' }*

PokerEvaluator.evalHand(["As", "Ac", "Qs"]);

*//{ handType: 2,*

*// handRank: 441,*

*// value: 8633,*

*// handName: 'one pair' }*

**2 + 2 Algorithm**

As mentioned, the hand evaluator effectively implemented the 2 + 2 algorithm. The algorithm calculates effective or *potential* hand strength. The formula is as follows:

EHS = HS x (1 –NPOT) + (1 – HS) x PPOT

EHS – Effective Hand Strength

HS – Current Hand Strength

NPOT – Probability that player hand deteriorates and becomes a losing hand.

PPOT – Probability that our current hand improves and becomes the winning hand.

Hand Strength (HS) will enumerate all possible opponent hand cards and count the occurrences where our hand is strongest (+50% of the cases where we are tied):

HandStrength(ourcards, boardcards) {

ahead = tied = behind = 0

ourrank = Rank(ourcards, boardcards)

for each case(oppcards) {

opprank = Rank(oppcards, boardcards)

if (ourrank>opprank)

ahead += 1

else if (ourrank==opprank)

tied += 1

else

behind += 1

}

handstrength=(ahead+tied/2)/(ahead+tied+behind)

return(handstrength)

}

In addition, EHS will consider the hand potential (i.e. its probabilities to improve or deteriorate):

HandPotential(ourcards,boardcards){

// Hand potential array, each index represents ahead, tied, and behind.

integer array HP[3][3] //initialize to 0

integer array HPTotal[3] //initialize to 0

ourrank = Rank(ourcards,boardcards)

//Consider all two card combinations of the remaining cards for the opponent.

for each case(oppcards){

opprank = Rank(oppcards,boardcards)

if(ourrank>opprank)

index = ahead

else if(ourrank=opprank)

index = tied

else

index = behind

HPTotal[index] += 1

// All possible board cards to come.

for each case(turn,river){ //Final 5-card board

board = [boardcards,turn,river]

ourbest = Rank(ourcards,board)

oppbest = Rank(oppcards,board)

if(ourbest>oppbest)

HP[index][ahead]+=1

else if(ourbest=oppbest)

HP[index][tied]+=1

else

HP[index][behind]+=1

}

}

//Ppot: were behind but moved ahead.

Ppot = (HP[behind][ahead]+HP[behind][tied]/2+HP[tied][ahead]/2)/

(HPTotal[behind]+HPTotal[tied])

//Npot: were ahead but fell behind.

Npot = (HP[ahead][behind]+HP[tied][behind]/2+HP[ahead][tied]/2)/

(HPTotal[ahead]+HPTotal[tied])

return(Ppot,Npot)

}

**Modeling Human Opponents**

Modeling human opponents was arguably the biggest challenge we faced. Correctly modeling a realistic opponent involves modeling a human’s mistakes as well as their successes. Computer implementations need to bluff and be able to respond to deception played by the opponent.

There are four types of strategies we attempted to implement:

*Loose Passive*: Typically less dangerous players. Always call, play a lot of hands, never raise or bet.

*Loose Aggressive:* Unpredictable yet exploitable. Tend to raise even with low hands.

*Tight Passive:* Never raise, play a few hands and they do they call.

*Tight Aggressive:* Play few hands, but when they do, they raise and raise again. Considered most dangerous category.

A good simulator would adapt its style of play to one of these depending on the game state, and would be able to change often enough that the opponent cannot pin-point and determine which style is used in certain situations.

**Poker Strategies**

We wanted to implement a few different poker strategies:

*Check-raise* – Used to intimidate opponent and banned in some casinos.

Player A checks

Player B Raise $100

Player A Raise $100 (Bet is now at $200)

*Continuation Bet* – Bet between 2/3 – ¾ of the pot. Great for low-level play.

*Semi Bluff ­–* Good for protection

Table: 2h 4c 5d

Hand: 6s Jd

*Total Bluff* – No protection

Table: 2h 4c 5d

Hand: Ts Jd

*Float Play –* Call on the flop, check-raise on the turn.

**Our Implementation**

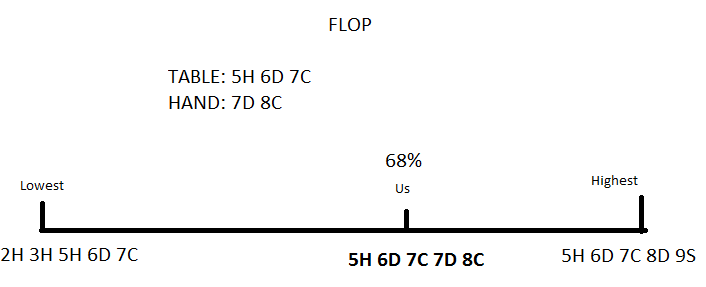
At the start of each round of betting round, we determine how aggressive our player should be and then, set a limit on how much they should bet for this particular hand. If playing aggressively, we will bet high in the hopes of scaring the opponent in to folding. If playing conservatively, we only bet high if we are confident we will win.

To determine if we should play aggressively, we keep a log of how many times the opponent folds. If an opponent folds more than 50% of the time, we will typically play more aggressive in hopes the opponent will continue to fold after bets are placed.

After it is determined how aggressive to play for this hand we then evaluate the hold cards. Our two-card evaluator returns the percentage they will end up with the winning hand. We use this percentage to determine how much money we are willing to bet for this hand.

For instance if the player has $1000, and has an initial hand strength of a 70, we would not want to bet more than $700 through out the entire hand. If the hand strength was 30%, we wouldn’t bet more than $300 before folding.

To advise betting post flop, we used the community cards to our advantage. Since our implementation only uses one deck of 52 cards, it was possible to do calculations on possible opponent’s hands. Our goal was to evaluate the lowest possible hand our opponent could have, evaluate our hand, evaluate the highest or best possible hand our opponent could have, determine a percentage of the player’s odds to win.



**Determining Best and Worst Opponents Hands**

To find the best possible hand, we run the community cards through a filter of the various hand types. The filters are ordered from the strongest hand types to the weakest. Since our implementation only uses one deck of 52 cards and has only two players, the opponent can have any two cards in their hand that are not community cards or hole cards of the player. Therefore, we check every combination of cards with the community cards on the table.

To determine the worst possible hand the player’s opponent can have involves low cards, no matches, not a straight and no flush. There were certain checks we used to find these. First, if there were any matches in the community cards, the opponent cannot have a straight. If there is a chance for a straight, we grab a card either two lower than the lowest community card or two above the lowest card is a 3 or 2. Finally, we check to see if there is a chance for a flush by looking at the suits available in the community cards, and chose the simulated cards’ suits accordingly.

**Other Simulator Techniques**

Artificial Intelligence researchers have been studying the game of poker as a whole for over a decade with vastly different theories. There have been implementations including expertly defined rules, machine learning, case-based reasoning, Monte Carlo methods, proving poker is a very complex game that many different methods could not fully simulate a human player. Many suggest multiple heuristic implementations that evolve over time are preferred, while others recommend neural networks because “they have been shown to be able to find solutions to non-linear search problems such as Poker” (Nicolai and Hilderman).

On September 5, 2013, the New York Times magazine published a six-page article on the *Texas Hold ’Em Heads Up Poker,* a limit version of the game that plays so well it can be counted on to beat players on almost every skill level. The machine players “paranoid” never wanting to exploit itself, therefore it follows a different strategy at each stage in every hand. The machine is fed data that has been mined from over 2-billion poker hands through neural networks and to present day has been beaten less than one-hundred times.

**Conclusion**

Human players have many advantages over a machine, like the ability to read and understand an opponent. However, there is a case to be made that machines can reduce the gap by quickly gathering statistical evidence that will deem them superior in other aspects of the game. This brings us back to the A.I. field in general, where the human mind has the ability to excel at aspects of problems that remain difficult to computers, and bridging that gap.

After analyzing just the basic complexity of the game and the various implementation methods that have been attempted, it is safe to say there are many benefits and disadvantages to each one. Although only one machine has been able to defeat the top players in the world, there is still significant progress being made. New methods are being enhanced every day, and that may someday, with more powerful hardware, have a chance at emulating and outperforming all human players.

Works Cited

Billings, D., Davidson, A., Schaeffer, J., & Szafron, D. (2002). The challenge of poker. Artificial Intelligence.

Bosilj, P., Palasek, P., Popvic, B., & Stefic, D. (2011). Simulation of a texas hold'em poker player.

Kaplan, Michaeel (2013) The Steely, Headless King of Texas Hold’Em.

Koller, D., & Pfeffer, A. (1997). Representations and solutions for game-theoretic problems. Artificial intelligence.

McCurley, Patrick (2009) An Artificial Intelligence Agent for Texas Hold’Em Poker.

Rubin, J., & Watson, I. (2011). Computer poker: A review. Artificial intelligence.