Poker Hand Strength

CGU MATH 387: Discrete Mathematical Modelling with Professor Allon Percus

Neel Adhiraj Kumar and Noah Keshner

December 15, 2017

Abstract

We describe a model for the game of Texas Hold'em Poker and a set of algorithms to evaluate the strength of hands at various stages of the game. For the purposes of this project, we do not use any external information (e.g., opponent's bet size, position at the table, opponent's game history), instead applying a general strength metric to a hand based only on the cards known to us. We use a combination of brute-force simulation and Monte Carlo estimation to estimate the probability of our hand winning the game. In addition, we compare the computationally demanding estimates of our starting hand strength to Chen formula scores - easy to calculate heuristic evaluations of hand strength designed for human players, and find that the two evaluations show a definite correlation. Combining results from our Chen formula scores and our probability estimations, we outline a more robust method for evaluating poker hands, which accounts for the probable strength of the opponent's hand as well.

1 Introduction

Texas Hold'em Poker is a card game with discrete states (52 distinct possible cards), stochasticity (each card drawn is probabilistic, not mathematically determined), and imperfect information (we do not know all game states - in particular, our opponent's hole cards). This makes it fertile ground for game theoretical analysis and artificial intelligence research, and various approaches have been developed to both aid humans playing the game and design successful artificial players [1] [2].

While a complete model of the game can be quite complex, we focus on the problem of estimating the strength of a Texas Hold'em Poker hand with limited information, assuming that an opponent's cards are unknown (as they would be in a real game of poker). Initially, we will assume our opponent is equally likely to have every possible set of starting cards, though a more complex model may adjust this assumption, using opponent modeling through either game theory or machine learning, or a combination of both, to estimate the opponent's hand more intelligently.

We compare our estimate to the Chen formula score of a hand – a (relatively) easy to calculate and popular estimate of the strength of a starting (pre-flop) hand. We also use our algorithm to try and identify especially strong and weak starting hands.

2 Model

A poker game starts out with each player being dealt two hole cards, information to which only that player is privy. After a round of betting, the flop of three community cards is revealed. Rounds of betting precede and follow the reveal of the fourth community card, the turn. And after the fifth and final community card, the river, is revealed, the last round of betting proceeds. If there are still multiple players in the game, the game reaches showdown: players must reveal their cards. The player with the best hand wins. If there is a tie, the pot is split evenly. There are several predefined patterns in poker that a player's cards, combined with the board (community cards), can form. A poker hand is judged first on the rarity of its pattern and second (if need be) on the magnitude of its highest rank.

We model the cards as integers (0-51) which allows us to conveniently read the rank (2,3,4,...,10, jack, queen, king, ace) and suit (clubs, diamonds, spades, hearts) of a card:

$$\text{rank} = \text{card} \pmod{13}$$
$$\text{suit} = \lfloor \frac{\text{card}}{13} \rfloor$$

The integer value of rank corresponds to its magnitude in the game (an ace has an integer value of 12, while a 2 has an integer value of 0). We do not bother to ascribe an integer value to a particular suit, only to distinguish between suits, as each is of equal weight in the game. This representation allows us to detect patterns based on rank and suit efficiently.

2.1 Hand Type

There are certain predefined hand-types in poker - patterns of cards of varying value. In order from highest valued to lowest valued, they are:

1. STRAIGHT FLUSH: Five cards in a sequence in the same suit
2. FOUR OF A KIND: Any four cards of the same rank
3. FULL HOUSE: Three of a kind and a pair
4. FLUSH: Five cards, not in a sequence, of the same suit
5. STRAIGHT: Five cards in a sequence, but not in the same suit
6. THREE OF KIND: Three cards of the same rank and two unmatched cards

7. TWO PAIR: Two cards of any one rank and two cards of another with an unmatched card
8. PAIR: Two cards of any rank and three unmatched cards
9. HIGH CARD: five unmatched cards

When we look for the best hand, we are trying to determine which player has the highest valued hand-type. Thus, the first step to evaluating the strength of a hand is detecting hand-type, (i.e. a set of predefined patterns in the hand). If two players have the same hand-type, the player with the higher ranked cards (especially those that are part of the pattern of the hand-type) wins.

2.2 Hand Comparison Implementation

We model Texas Hold'Em Poker's hand comparison using the following functions:

2.2.1 Hand Type

Given an input of five unique cards this function outputs an integer value from 0-8 corresponding to the cards' hand-type, with higher magnitudes representing stronger hand-types. In our case, N is equal to the lengh of the hand we are examining, which for our purposes will always be 5.

Complexity: $O(N)$

2.2.2 Hand Ranks

This function simply takes a list of integers corresponding to different cards and outputs a list of their ranks in descending order. This allows us to efficiently compare two hands when they have the same hand-type.

Complexity : $O(N \log N)$

2.2.3 Compare

This function takes two hands (in the form of a list of integers) as input, and outputs an integer corresponding to whether or not the first hand won, lost, or tied (according to the hand-type and ranks). Because this is naturally a function of the other two functions, its complexity is astronomically larger. This might be worrisome, but in practice we are only dealing with $N = 5$, corresponding to the length of each hand, so the computation is quite manageable.

Complexity : $O(N^4 \log N)$

2.2.4 Best Hand

At showdown, each player has seven cards available to them: their two hole cards, and the five community cards on the board. Thus, the final element needed for comparison of two hands is a function that identifies the best five-card combination from each. Given up to seven cards, this function returns the strongest hand out of every five-card combination. To do this, it runs the *compare* function on two different combinations at a time and maintains the best one, for all $\binom{7}{5} = 21$ possibilities. While this problem would quickly become intractable for large N, we have the privelege of knowing that $N \leq 7$ in every applicable scenario.

Complexity : $O(N!N^4 \log N)$

3 Hand Strength

A necessary part of playing poker is accurately evaluating the strength of a given hand. Though more advanced strategies (particularly involving betting) follow from this, players need to have an approximate understanding of the underlying strengths (or weaknesses) of their cards to even begin developing a strategy. All other things equal, a player that can intelligently read his or her chances of winning a hand will have perform more successfully than one who cannot. In this regard, a computer has a significant advantage over a human player, as it has the ability to quickly calculate complex probabilities that a human feasibly could not. However, in a real game of poker, understanding the odds is not as simple as calculating strict, static probabilities. Unless one's opponent is operating under rules which are random or oblige him or her to call every bet, the assumption that an opponent is at every stage of the game equally likely to be holding a weak hand as a very strong one is faulty. Hand strength calculations will minimally depend on our own cards and the cards on the board, but more ambitious models can be extended to include information regarding our opponent's cards (for instance, betting history) as well. [3]

Nonetheless, this rather naive approach to evaluating hands (which assumes that the combination of an opponent's cards follows a uniform distribution over all possible hands) is an informative starting point, and we will first outline a method of evaluation under such an assumption. Though these probabilities will certainly not represent the true chances that a player wins a hand, the values that our model outputs will have a significant and strong correlation with the utility of a hand in any scenario. Afterward, we will propose a few mechanisms which can be applied on their own or in conjunction to reduce the naiveté of these calculations.

4 Algorithms to Estimate Hand Strength

Now that we have established functions to systematically compare known poker hands, we can begin to explore methods for evaluating hand strength when we do not have the whole picture. We use different techniques in different situations, depending on the amount of board information provided to us.

4.1 Post-flop: Naive Brute-force Simulation

In the post-flop case (after at least three community cards have been revealed on the board), to estimate the strength of our hand, we use an algorithm that enumerates all possibilities for all hidden game states, and we count what percentage of those we win (plus half the percentage of those we tie).

Hence, we draw every possible remaining community card, and for each of those possibilities, consider every possible set of opponent hole cards. The enumeration is feasible, though at the flop we are already considering over a million possibilities.

River: $\binom{45}{2} = 990$ opponent card possibilities (.5 seconds to calculate on average)
Turn: $\binom{46}{1}\binom{45}{2} = 45,540$ opponent card possibilities (7 seconds to calculate on average)
Flop: $\binom{47}{2}\binom{45}{2} = 1,070,190$ opponent card possibilities (8 minutes to calculate on average)

4.2 Pre-flop: Monte Carlo Sampling

In the pre-flop case, if we were to enumerate every possibility, the number of cases to check would be unmanageably large:

$$\binom{50}{5}\binom{45}{2} = 2,097,572,400$$

Instead of using a brute-force simulation here as in the post-flop cases, we use Monte Carlo sampling to reliably estimate the distribution of cards on the board and with our opponents. With our given starting hand, we play N games to completion, and calculate the fraction of games we win (plus half the games we tie).

On modern computers, this algorithm has feasible average time performance:
N = 100,000: 132 seconds
N = 10,000: 13 seconds

4.3 Pre-flop: Chen Formula

The Chen formula is a strategy developed for human players by William Chen to heuristically evaluate their starting (pre-flop) cards [4]. The Chen formula is a series of simple tests and calculations to generate a score for each set of hole cards. Each test tries to account for the potential strategic value of the cards, and give it a corresponding score. The Chen point count of a starting hand corresponds quite closely with the Sklansky and Malmuth hand grouping, another set of (relatively) simple strategies designed for human poker players. Hence, we can consider this an approximate, game theoretic evaluation of hand strength [4].

The following steps outline the Chen score calcuation, with the score being initialized at 0 and being updated accordingly at each step:

1. Score your highest card only. Do not add any points for your lower card.

   A = 10 points.

   K = 8 points.

   Q = 7 points.

   J = 6 points.

   10 to 2 = $\frac{1}{2}$ of card value. (e.g. a 6 would be worth 3 points)

2. Multiply pairs by 2 of one card's value. However, minimum score for a pair is 5.

   (e.g. KK = 16 points, 77 = 7 points, 22 = 5 points)

3. Add 2 points if cards are suited.

4. Subtract points if their is a gap between the two cards.

   No gap = -0 points.

   1 card gap = -1 points.

   2 card gap = -2 points.

   3 card gap = -4 points.

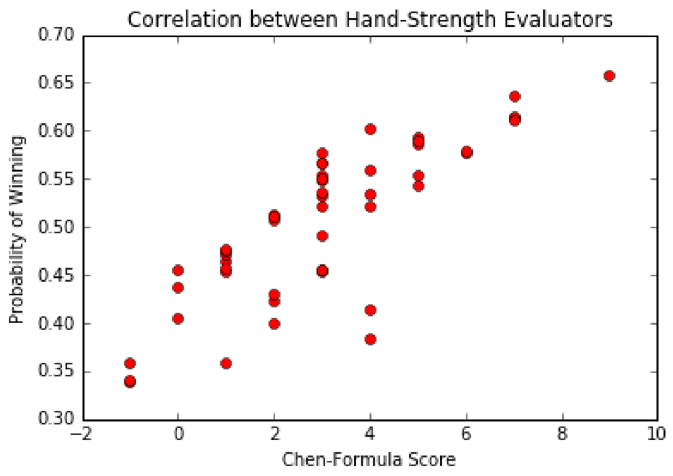
   4 card gap or more = -5 points. (Aces are high this step, so hands like A2, A3 etc. have a 4+ gap.)

5. Add 1 point if there is a 0 or 1 card gap and both cards are lower than a Q. (e.g. JT, 75, 32 etc, this bonus point does not apply to pocket pairs)

6. Round half point scores up. (e.g. 7.5 rounds up to 8)

4.4 Correlation Between Evaluators

After 50 trials, we find that our Monte Carlo percentile estimates of hand strength (with $N = 100,000$) correlate somewhat with Chen-formula scores for starting hands, though of course there is far less room for variation in the latter.



Starting Hand	Estimated Hand Strength	Chen Score
AA	0.854	20
KK	0.812	16
QQ	0.778	14
JJ	0.746	12
TT	0.719	10
99	0.689	9
AK (same suit)	0.674	11
AQ (same suit)	0.670	10
AJ (same suit)	0.661	8
88	0.660	8
AK (off suit)	0.659	9
AT (same suit)	0.654	7
AQ (off suit)	0.651	8
AJ (off suit)	0.642	6
A9 (same suit)	0.639	7

Table 1: Scores of the best starting hands.

4.5 Reducing the Naiveté of the Model

To this point, our estimated hand strength has been naive in the sense that it assumes the underlying distribution of opponent cards is uniform. As mentioned before, even the most inexperienced players will play strong hands differently than weak hands. While a number of different factors should ultimately be considered in order to accurately estimate the opponent hand distribution, one tenable method for estimating hand strength under this distribution is to weight strong starting hands more heavily than weak starting hands in our probabilistic samples. For instance, we know that a player with a pair of aces as hole cards is much more likely to stay for the entire game than a player with a couple of unsuited, unpaired, low cards.

Conveniently, the Chen formula provides an efficient method for scoring starting hands. We can now apply a weight according to $\max\{\text{score}, 1\}$ for each possible opponent hand in our original algorithm. Thus, the adjusted algorithm runs as before, altering only three steps in the initial naive process:

1. Calculate the sum of all $\max\{\text{score}, 1\}$ for each potential opponent hand.
2. For each win or tie, add $\max\{\text{score}, 1\}$ to the tally of wins and ties (as opposed to just 1).
3. Divide by the this tally of wins and $\frac{\text{ties}}{2}$ by the sum calcuated in the first step.

This addition to the algorithm recognizes that an opponent is more likely to be participating in a hand if he or she has good hole cards, and that the stronger these cards, the more likely an opponent is still in the hand. The change in starting hand rankings from Table 1 to Table 2 reflect this change. In particular, middling pairs are shown to be significantly weaker starting hands than our previous algorithm suggested, due primarily to the prevalence of strong hands containing higher ranked cards. Further, the supremacy of pocket aces is accentuated even more by this algorithm, showing a nearly 9% differential in likelihood of winning a hand between pocket aces and pocket kings, the second best starting hand. From this new table of top 20 hands alone, it is already apparent that the algorithm is operating under conditions which more accurately model opponent hands, as these updated evaluations more closely resemble the relative values a professional human player would assign to them.

Starting Hand	Estimated Hand Strength	Position Change
AA	0.846	-
KK	0.756	-
QQ	0.707	-
JJ	0.669	-
AK (same suit)	0.655	$\triangle + 2$
AQ (same suit)	0.637	$\triangle + 2$
AK (off suit)	0.636	$\triangle + 4$
TT	0.636	$\triangledown - 3$
AJ (same suit)	0.620	-
AQ (off suit)	0.618	$\triangle + 3$
AT (same suit)	0.608	$\triangle + 3$
99	0.603	$\triangledown - 6$
AJ (off suit)	0.599	$\triangle + 1$
A9 (same suit)	0.588	$\triangle + 1$
88	0.579	$\triangledown - 5$

Table 2: Adjusted scores of the best starting hands.

5 Algorithm Comparison

To attain a fuller illustration of the discrepancies between our naive algorithm and the adjusted algorithm, we examine a real-life Texas Hold'em hand between Patrik Antonius and Marco Graziano in the 2017 Pokerstars Championship. In this particular hand, Antonius was dealt pocket queens ($[Q\clubsuit, Q\diamondsuit]$), while Graziano held pocket sixes ($[6\spadesuit, 6\heartsuit]$). However, for the purposes of this test, we will assume Graziano's hand is unknown and observe how each algorithm performs at all stages in the game from the perspective of Antonius.

5.1 Pre-Flop

Given that Antonius holds queens, and both algorithms rate it as the third best starting hand in Texas Hold'em, there would be essentially no reason for Antonius to fold at this time. However, while the naive algorithm is incredibly bullish on Antonius's chances, rating them at 77.8%, the adjusted algorithm is much more conservative, estimating a 70.7% chance of victory. In either case, with no other information, it is a straightforward decision not to fold.

5.2 Flop

The flop reads: $[A\heartsuit, J\clubsuit, A\clubsuit]$. This merit of this flop in relation to our starting hand, ($[Q\clubsuit, Q\diamondsuit]$), is relatively ambiguous to an untrained human eye. All in all, it seems like a positive step forward for the pocket queens, being that the current hand-type is upgraded to a two-pair, and the queen of clubs provides the opportunity for a flush. In addition, the fact that Antonius has two queens make it less likely that an opponent will be able to complete a straight (as the chance of an opponent holding a queen is significantly reduced). On the other hand, if an opponent has an ace, this hand falls far behind.

Naive Estimation: 0.822
The naive algorithm calculates an increase in value after this flop, as the two aces on the board give fewer winning combinations to the opponent. Though this is a somewhat reasonable estimate, it seems to give very little credence to the fact that an opponent might be holding an ace, in which case a loss is almost certain.

Adjusted Estimation: 0.695
Unlike the naive estimator, the adjusted algorithm attaches a lot more weight to the potential for an opponent to have an ace, and as a result the value of Antonius's starting hand, while still greatly positive, decreases after the flop. This makes sense since an opponent with a weak hand is more likely to have folded in the round of betting preceding the revealing of the flop, whereas an opponent with an ace (a strong card in any case) is more likely to remain.

5.3 Turn

After the turn, the board reads: $[A\heartsuit, J\clubsuit, A\clubsuit, K\spadesuit]$. This is not a positive card for Antonius, as it is completely feasible that Graziano is holding a king. Additionally, though Antonius's two queens protect him significantly from this possibility, Graziano now has the potential to be holding a completed straight.

Naive Estimation: 0.787
Predictably, following the turn of the king, Antonius's hand strength estimate drops from the flop. However, given how high his predicted chances were immediately after the flop, the decrease is relatively insubstantial. His chances of winning at showdown are rated to be even higher now than they were at the very start of the hand, despite some very dangerous cards for him on the board.

Adjusted Estimation: 0.646
The adjusted estimator suggests that this hand has become weaker than it was both at the start and after the flop. As before, because opponent hands containing kings and aces are considered more likely (either of which would beat us), this estimator is more conservative in evaluating the worth of pocket queens in this position.

5.4 River

Finally, follwing the river, the board reads: $[A\heartsuit, J\clubsuit, A\clubsuit, K\spadesuit, J\diamondsuit]$. This, again, is not an innocuous card for Antonius. Now, if his opponent holds an ace, a king, or a jack, he loses the hand - all of which are well within the realm of feasibility.

Naive Estimation: 0.702
The naive estimate of Antonius's hand strength drops by over 8%. However, it still rates Antonius's chances of winning at over 70%. In almost all cases, poker players who believe they have a 70% chance or greater of winning a hand will be willing to bet a lot on their cards. But, given our increasing concerns with the board, this is definitely not a clear call to make, as the naive estimator's prediction might suggest.

Adjusted Estimation: 0.558
With the pairing of the jack on the river, the adjusted estimation of Antonius's hand strength falls by an even greater magnitude than the naive estimation, despite already starting at a significantly lower point. Though his chances of winning are still valued positively, the adjusted estimation reflects the reality that this is a tough call - nearly a coin flip.

5.5 Conclusion

As both our algorithms correctly suggest, Antonius called Graziano's bluffs and took down the pot, albeit after considerable deliberation. This was by no means an easy call, and we can see from this example that the adjusted estimator can at least in certain cases provide a more accurate estimate of the strength of a hand than the naive one. Intuitively, it seems that an opponent with Aces, Jacks, or Kings is more likely to remain for the duration of the game than a player with a weaker hand, a reality incorporated in our adjusted model but absent from the naïve one.

6 References

1. D. Billings, D. Papp, J. Schaeffer, D. Szafron, Opponent modeling in poker, in: Proc. AAAI-98, Madison, WI, 1998, pp. 493–499.
2. Teófilo, L.F.: Estimating the Probability of Winning for Texas Hold'em Poker Agents. In: Proceedings of the 6th Doctoral Symposium on Informatics Engineering, pp. 129–140 (2011)
3. Rubin, J., Watson, I.: Computer poker: A review. Artificial Intelligence 175(5-6), 958–987 (2011)
4. Chen, B., Ankenman, J.: The Mathematics of Poker. Conjelco (2006)