Winning Opponent Counter Strategy Selection in Holdem Poker

Nathan Lloyd
Department of Computer Science
Cornell University
Ithaca, NY 14853
nsl6@cornell.edu

Abstract

The game of poker presents an interesting and complex problem for game theorists and researchers in machine learning. Current work on the subject focuses on how to develop optimal counter strategies, often referring to the Upper Confidence Bounds (UCB1) algorithm to determine which of these counter strategies is optimal for an unknown opponent. We present a new method for taking a learned set of counter strategies and selecting a winning strategy to employ against an unknown opponent.

1 Introduction

In 2003, a previously unknown man named Chris Moneymaker qualified for the $10000 World Series of Poker main event in a $40 qualifier tournament, and went on to win the biggest prize in poker, sparking an explosion in the popularity of the game. In the academic world of game theory, artificial intelligence, and machine learning, this ignited an interest in a new research area: artificially intelligent poker agents.

The most popular variant of poker played today is No Limit Texas Holdem. In this variation, each player is dealt two hole cards and five cards in total are used as community cards which each player can use to try to make the best five card poker hand. Due to the number of betting possibilities open to players at every decision, the growth of the extensive game tree for No Limit Texas Holdem is astronomical, easily swelling to over 10^{20} nodes.

This has created a flurry of interest among researchers, as nobody can yet claim to have solved poker. Solving the complex game is a matter of combining a collection of known strategies with a method of selecting a winning strategy to employ against an unknown opponent to maximize your net money won.

Extensive research has been done on the calculation of optimal counter strategies in a relatively small amount of time; it is the second segment of the problem that is of particular interest. Given a collection of optimal counter strategies for known, fixed strategy opponents, how can we efficiently select the strategy to employ against an unknown opponent during a live match?

2 Previous Work

Much of the current literature on artificial poker research focuses on counter strategy development, while the method of selecting which strategy to employ against an unknown opponent is often not the topic of discussion. The selection algorithm often used by entrants in the AAAI Poker Competition is the UCB1 algorithm. [1] UCB1 computes upper confidence bounds for each strategy as a function of the average payout and the number of times that strategy has

been used. The equation is shown below, where $\bar{x}_j$ is the average payout for strategy j, n is the total number of hands played, and n_j is the number of hands played using strategy j.

$$Strategy = \underset{j}{argmax}\left(\bar{x}_j + \sqrt{\frac{2\ln(n)}{n_j}}\right) \quad (1)$$

The benefit of the UCB1 algorithm is that it is guaranteed to converge to the optimal strategy in your strategy set. If a Nash Equilibrium strategy is known, after convergence you will never have a losing strategy. However, there are two major pitfalls in the UCB1 algorithm as applied to poker. First, there is no bound on the convergence period. A number of factors contribute to the convergence rate, including the number of counter strategies in your set and the unknown opponent's strategy. Secondly, UCB1 assumes that the opponent's strategy is fixed at all times. It is easily exploited if the opponent switches its strategy in the middle of a match. For example, if one agent uses UCB1 converges to the optimal counter strategy after 50000 hands and continues to play that strategy only, the opponent can switch its strategy at hand 60,000 and the agent using UCB1 may never recover from this switch.

3 Classified Counter Strategy Selection

Given these deficiencies in the UCB1 algorithm, we developed a method of selecting a counter strategy that sacrifices optimality for efficiency in convergence. This new approach, called Classified Counter Strategy Selection, uses a supervised learning algorithm to select the most appropriate strategy based on the attributes that make up a poker player. These attributes populate a vector of well known poker statistics. Some examples of attributes contained in this vector are: VPIP, the percentage of hands the player voluntarily put extra money in the pot before the flop; PFR, the percentage of hands the player raised before the flop; and the aggression factor, a numerical measure of how aggressive a player is. All attributes are normalized to be a decimal value between 0 and 1.

Each counter strategy in our set is associated with a label in our learning algorithm. To train our classifier, we take unknown opponents whom we have an extensive number of hands recorded and solve for the expected value of playing each of our counter strategies. The label of the counter strategy with the largest expected value is the label we put on that opponent's attribute vector. These attribute vectors effectively describe the classes of opponents that each counter strategy is effective against rather than associating each counter strategy with a specific opponent. During a match with an unknown player, we choose the counter strategy which best exploits the class of player we believe this new opponent falls into.

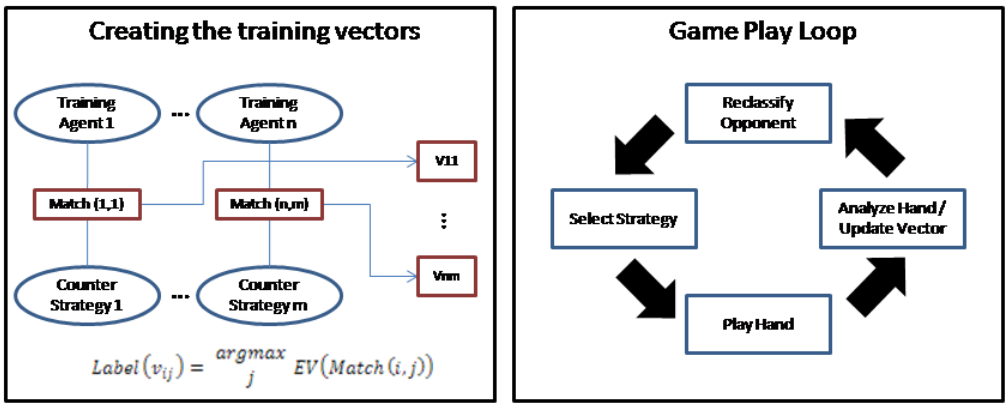

Figure 1: Diagram of the complete learning process used in the Classified Counter Strategy Selection algorithm

Classified counter strategy selection can be efficiently computed in a negligible amount of time between each hand in a match. After a hand is completed we analyze it and update the attributes in the opponent's vector. Following this update, we reclassify the opponent and employ the associated counter strategy against the opponent on the following hand. This algorithm provides potential benefits for both of UCB1's pitfalls:

1. **Convergence Rate**: Early in the match, updates to the attribute vector will cause major fluctuations and strategy selection may be very erratic. This is due to the size of the game tree, and the fact that a small number of hands simply don't cover enough scenarios in a poker game to be useful. However, as the match progresses, the attribute vector converges, as shown in figure 2 for varying simple opponent strategies. Since the vector effectively converges, the counter strategy selected from hand to hand will remain constant. Since the vector converges, strategy selection is guaranteed to converge.

2. **Non-Fixed Opponent Strategies**: Due to the quick convergence rate of the algorithm, opponents that switch their strategy in the middle of the match will no longer be a problem. In early experiments, we found that the attribute vector often converged in as little as 100 hands. Knowing this, instead of using all hands in a match, we can instead only update the vector with the latest n hands, called windowing. Now we can converge to the new attribute vector within n hands of the opponent's strategy change.

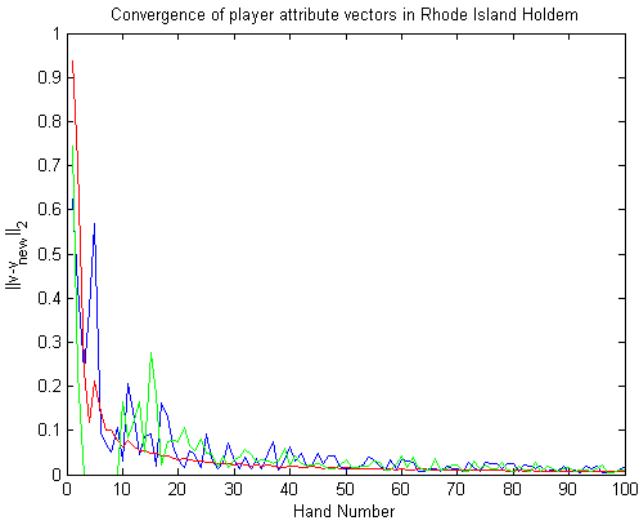

Figure 2: Convergence of the player attribute vectors for an agent that calls at every decision (red), is completely random (green), and one that uses an even probability distribution at each decision (blue)

.

Two machine learning algorithms were used in testing the opponent vector approach. The first was a traditional K-Nearest Neighbor (KNN) implementation and the second was Thorsten Joachims' implementation of a multiclass Support Vector Machine (SVM). Early in testing we compared the SVM implementation and the KNN implementation against the UCB1 algorithm in order to get an idea of how the opponent vector classification approach would perform given the method of labeling. Figure 3 shows an example of the early results that we obtained for the two approaches against a single opponent. The SVM algorithm often resulted in a poor performance when testing (likely due to over fitting the training data), leading to use of the KNN algorithm when labeling our opponent vector and picking a counter strategy. A k value of 3 was selected for use in the KNN algorithm for similar reasons. This provided quick convergence without degrading the selection performance.

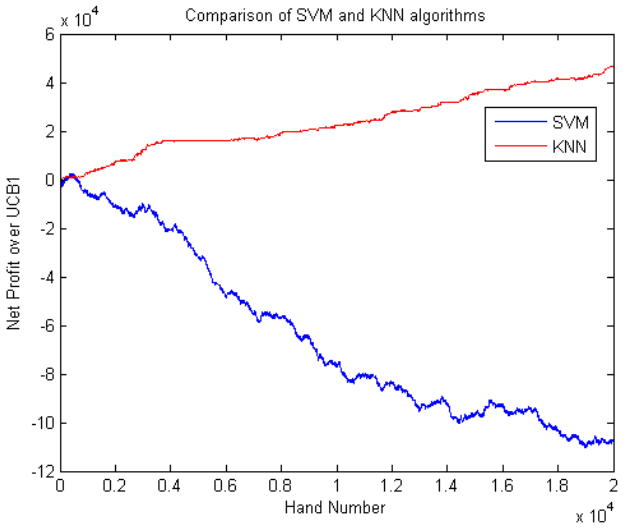

Figure 3: Comparison of using KNN (red) for vs. SVM (blue).

4 Experiment

4.1 Experimental Setup

A smaller, more manageable version of poker is called Rhode Island Holdem. This variation uses one hole card and two community cards to form a three card poker hand. Unlike No Limit Texas Holdem, Rhode Island Holdem is a limit poker game, meaning you must place bets in a specified increments rather than having the ability to risk all of your money at any point. We performed our experiments on this smaller poker game, which serves as a great testing bed for the classified counter strategy selection algorithm. Rhode Island Holdem gives us the ability to run the algorithm on a game which we can explicitly compute the entire game tree rather than worrying about the consequences of abstraction.

All data for our experiments was generated in house specific for this project. We wrote a collection of rule based agents, as well as some that used naïve attempts to exploit an opponent. These agents were divided into three categories. The first group is comprised of the counter strategies that the Classified Counter Strategy Selection agent and the UCB1 agent could select from.

	Training 1	Training 2	Training 3	Training 4	Training 5	Training 6
Strategy 0	**1.7956**	2.4844	1.2402	0	0.7482	**-0.3604**
Strategy 1	1.1564	2.1708	2.4608	0.3912	2.2836	-1.002
Strategy 2	0.6624	**4.9624**	**3.8628**	-0.7542	**4.0864**	-1.864
Strategy 3	1.5448	-0.3986	0.9318	**0.8458**	0.6798	-0.834

Figure 4: Results of training matches between the agents in the training set and each of the strategies used in the strategy set. The value shown is the number of betting units that the strategy won or lost on average per hand.

The second group was used for training. Each of these agents were matched against then agents in the first group to generate training vectors, as shown in figure 4. The value provided is the average number of betting units won per hand, where the ante is considered one betting unit, pre

flop bets are two units, and post flop bets are four units. We label the attribute vectors gathered for each training agent according to the associated strategy that had the highest average win rate against it. For example, strategy 0 had the highest win rate against training agent 1, so the attribute vectors recorded for training agent 1 are labeled with class 0.

The framework in which we execute our tests in is designed specifically for direct comparison of two strategies. We can deal out an n handed match and save the cards dealt to reuse in future matches. This allows us to test two different strategies against the same opponent without the randomness of the cards playing into the effect of the outcome. Additionally, we can take two opponents and swap their seats such that they see an identical set of hands from the opposite view. Unlike a human opponent, who will retain knowledge of the hands played, we can clear out the memory of the agents and start fresh for each match. This helps to measure whether one opponent is truly better than the other since they will play the same set of n hands from both positions.

4.2 Evaluation

The assumption made in our approach is that the attribute vector accurately summarizes an opponent strategy in such a way that we can select a counter strategy that performs well against a certain class of opponents. We are sacrificing optimality in strategy selection for a more efficient convergence rate. Generally, it is important to evaluate the performance of strategy selection on these three aspects:

1. The rate the algorithm converges to a learned counter strategy given a new opponent
2. The win rate before convergence
3. The win rate after converging to a learned counter strategy

Evaluation for these experiments is done by direct comparison with the UCB1 strategy selection algorithm, regardless of the actual net win/loss. Specifically: we look to have a better average win rate against an opponent than what the UCB1 algorithm can achieve when both strategy selection algorithms are using the same set strategies on identical sets of poker hands.

Theoretically, performance better than UCB1 can only be achieved by either selecting the optimal strategy quicker than UCB1 does or by choosing a strategy which has a net win over UCB1 before UCB1's convergence which is larger than the net win UCB1 has over the suboptimal strategy after convergence. Effectively, the longer the match, the more time we allow UCB1 to converge and ultimately perform better. Since we can never beat UCB1 after converging, we must beat it in the first two aspects.

4.3 Results

Selecting the appropriate one of four counter strategies against an unknown opponent is critical in achieving the highest win rate you can. Figure 5 shows an example match which the opponent attribute vector approach performed exceedingly well compared to the UCB1 algorithm. The average win rate for the new approach was 0.08 betting units higher than that of the UCB1 algorithm.

Note that UCB1 never converged to an optimal strategy because the number of hands in the match was not large enough; however, you can see progress being made toward convergence as counter strategy 1 seems to have been picked by UCB1 at the 10000 hand mark. The new approach quickly converged to use counter strategy 0, which may be a suboptimal choice. By quickly converging to a relatively good strategy, we were able to achieve a higher win rate than that of UCB1.

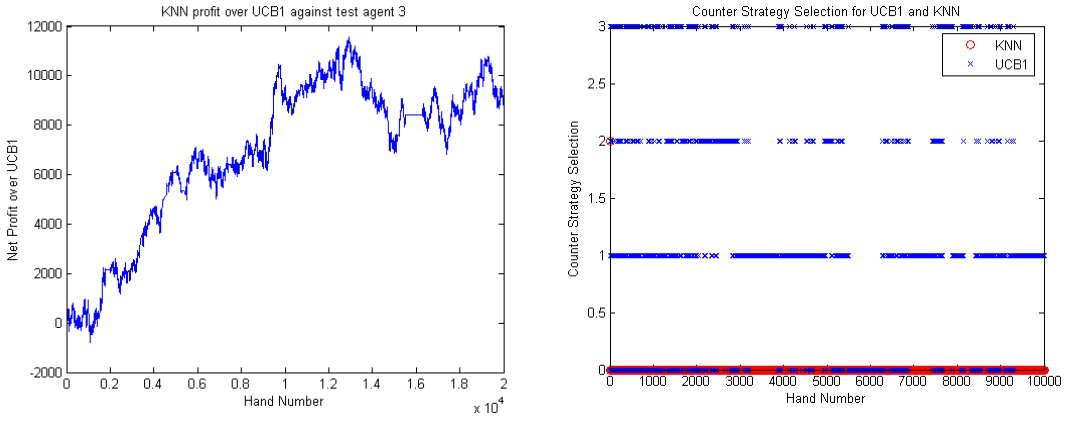

Figure 5: Left: The net profit of the KNN classifying strategy selector over the UCB1 algorithm against one of the test agents. Right: The strategy selection for each of those algorithms during that match.

A problem previously identified with the UCB1 algorithm is the possibility of an opponent changing their strategy in the middle of the match, making the data gathered thus far nearly useless. This is a critical technique for professional poker players to avoid exploitation, and one that should be emulated by artificial poker agents. Figure 6 shows the result of a match between each strategy selection method and an agent that exhibits this behavior, changing its strategy every 200 hands. In addition to the standard classified counter strategy selection algorithm, an agent that uses windowing was matched up against this opponent. Windowing allows the agent to recognize if an opponent changes their strategy by only using a certain number of recent hands to determine the counter strategy that is best to employ.

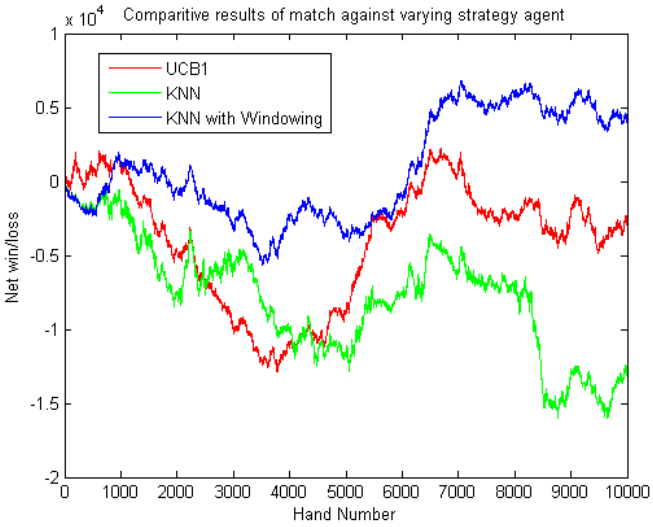

Figure 6: Net win/loss of the three strategy selection algorithms against the varying threshold agent.

As you can see, UCB1 did not fare well against this opponent, with an average loss of 0.05 units per hand. The standard classified counter strategy selection algorithm performed worse, losing 0.25 units per hand. However, once windowing is used, we achieve an average win of about 0.1

units per hand. This exemplifies the benefit of the new approach over UCB1. We cannot implement a similar windowing feature in the baseline because of the extended convergence period. If we converge to a strategy with UCB1 and the opponent suddenly switches to a new strategy, it will take an enormous number of hands before the algorithm converges to the new optimal counter strategy, and at that point it is likely that the opponent would have again switched.

A table of win rates for both of the approaches is provided in figure 7. Overall, the new approach outperformed UCB1 in 64% of the matches, with an average profit over UCB1 of 0.052 units per hand.

	Test 1	Test 2	Test 3	Test 4	Test 5
Classified Counter Strategy Selection	**-0.36165**	**0.1257**	0.7471	**0.03**	-0.4534
UCB1	-0.81405	0.0375	**0.95575**	-0.43645	**0.29315**

Figure 7: Table of average win rates against various opponents. Each match consisted of two 10000 hand matches where the opponents swap seats and clear memory halfway.

The final experiment performed was a match between the baseline approach and the new opponent vector approach. This match was performed just as all test matches were, duplicating the hands and swapping the hole cards of the two agents in the middle after clearing their memories. The result of the match was a little surprising given the results of the previous tests. The UCB1 algorithm came out a clear winner over the opponent vector approach both with and without windowing. This is likely because the UCB1 algorithm never converges to an optimal strategy in the limited number of hands, so its choice of strategy fluctuates nearly every hand. This is a problem for the opponent vector classification and, as a result, the vector cannot converge, so the advantage is loss.

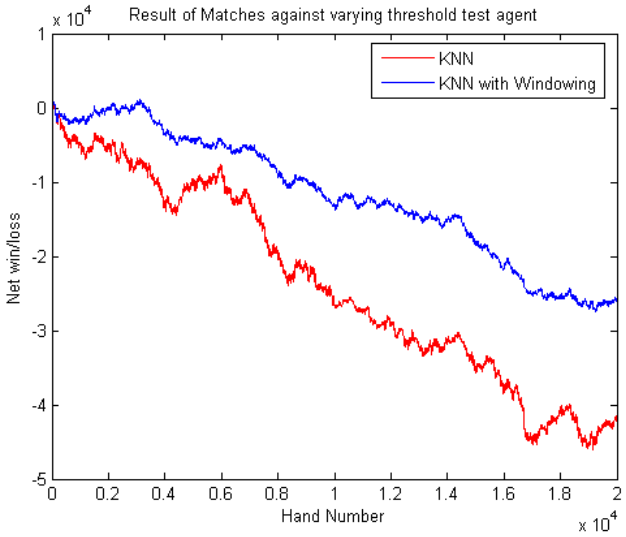

Figure 8: Net win for the opponent vector strategy selection algorithm when playing against the UCB1 strategy selection algorithm.

5 Future Work

With the promising results from Rhode Island Holdem, we can make the necessary changes to allow the algorithm to be run in a No Limit Texas Holdem environment. This will mainly involve reconstructing the opponent vectors to include attributes specific to that game, as well as developing more rule based agents and counter strategies specific to No Limit Texas Holdem.

The result from the match between the UCB1 agent and the classified counter strategy selection agent was eye-opening, and warrants further investigation when employed in the No Limit Texas Holdem game. Since many of the top tier implementations use the UCB1 algorithm, this approach must be robust enough to perform well against that algorithm.

We feel that the results found here will closely match the results we will find when testing our algorithms on No Limit Texas Holdem, resulting in an agent comparable to those who competed in the 2010 AAAI Poker Competition.

6 Conclusion

Classified counter strategy selection shows some promising results when compared to the baseline UCB1 method. For many of the test cases, we found that the UCB1 algorithm had a lower win rate due to the amount of time it took to converge to the optimal result. In matches where we quickly find the optimal strategy, the opponent attribute vector approach outperforms the UCB1 algorithm. Using the opponent vector we achieve convergence quickly; however, the algorithm can miss the mark and choose a suboptimal strategy, occasionally resulting in a lower win rate in certain matches.

Glossary of Poker Terms

Action: The opportunity to check, call, bet/raise, or fold.

All-in: When a player bets the amount of chips he/she has remaining.

Ante: A forced bet that all players must place before being dealt into a hand.

Card Rank: The numerical value of the card (2-10) or the title jack, queen, king, and ace.

Check: To pass the action to the opponent without betting.

Community Cards: The cards made available for all players to use.

Heads-up Play: A one-on-one poker game.

Hole Card(s): The card(s) dealt to each player that only the player can see.

Flop: The first community card(s) dealt after the conclusion of the first round of betting.

Pot: The total amount of chips bet by all player during a hand.

River: The final community card dealt.

Showdown: After the final round of betting, it is when both players reveal their hands to determine the winner.

Turn: The community card that follows the flop.

References

[1] Michael Bradley Johanson. *Robust Strategies and Counter-Strategies: Building a Champion Level Computer Player*. 2007. University of Alberta.

[2] Darse Billings. *Computer Poker*. 1995. University of Alberta.

[3] Michael Johanson, Martin Zinkevich, and Michael Bowling. *Computing Robust Counter-Strategies*. 2007. University of Alberta.

[4] Thorston Joachims. *SVM*^{multiclass}- Multi class support vector machine implementation. *http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html*