

Predicting Texas Holdem Hand Strength

James Bensson, Alex Eckert, Maxwell Wu

December 13, 2013

1 Motivation

It all comes down to this, you are in the final round at the Texas Hold'em World Series of Poker Championships. Just two people remain, you and Jack Manningham, the "river runner," who just went all in. What should you do, call him and try to end the game, or play it safe and let him steal the pot? Dilemmas similar to this constantly plague Texas Hold'em players and in order to help answer this question we developed models to predict a player's hand strength based on their betting behavior.

2 Variable Formalization

This paper will assume some basic knowledge of Texas Hold'em terminology. We were first presented with a large amount of unformatted data and had to decide on what to extract for use.

Target Variable - Hand Rank (HR)

Hand rank is defined as the percentage of other combinations of two hole cards that a player's current two hole cards will beat in hand strength. Hand rank does not take into account future expectations. Since poker hand rankings are linear, a higher hand rank strictly implies the hand will win versus another hand of lower hand rank. Knowing an opponent's hand rank on the river would thus allow a player to play with perfect information.

Figure 1 shows an example of a hand with a hand rank of .838. Although the player paired with the highest card on the board will beat many other hands, there are still many hands that can beat the player's hand, such as two pairs, triples, and straights.

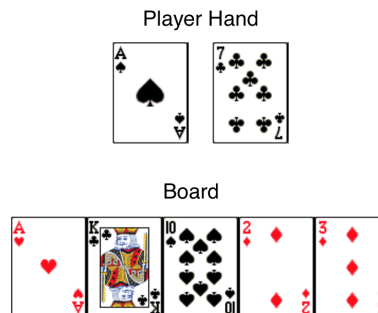


Figure 1: Example Hand Rank

Features

Initially, we used the following feature set for each player:

- number of checks before the river
- number of bets before the river
- number of calls before the river
- number of raises before the river

As we progressed in our analysis we chose more robust feature sets.

- Value of bet compared to the current pot amount
- Number of bets in certain ranges compared to the current pot
- Ratio of aggressive (bet/raise) moves to passive (check/call) moves
- Average bet in the round compared to the pot size
- Number of actions in a certain round

3 Data

We initially leveraged an online poker framework to generate player data, which allows users

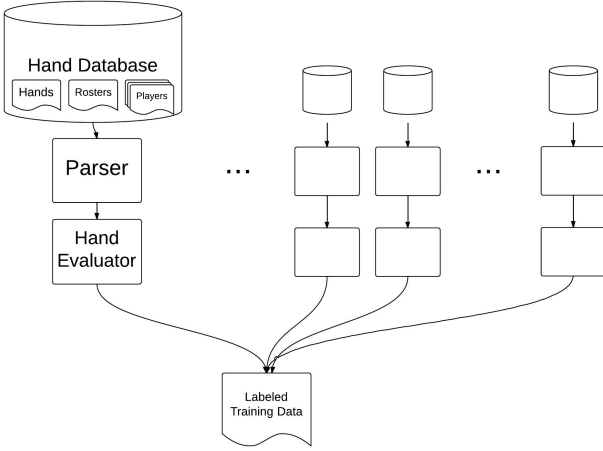


Figure 2: Data Parsing Pipeline

to create their own poker robots, alter existing poker robots, and play different robots against each other. The main benefit of initially using bot-generated data was transparency. No matter what happens in a round, we see the bots hand. Using bot data allowed us to get a sense for which algorithms would work well, but was ultimately not too interesting and did not accurately mimic real world poker conditions.

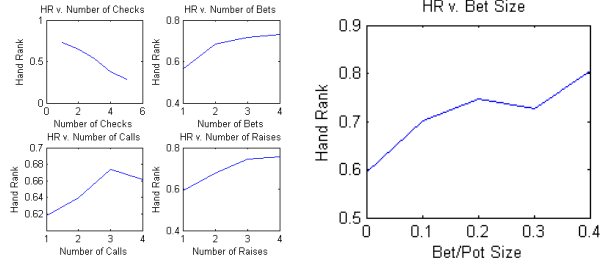
We switched to using a dataset of human poker games, provided by the University of Alberta Poker Research Group. The data set is of poker games played over IRC (by humans) and each player has a limited amount of chips. Because there is a scarcity of chips, players play to maximize their winnings and their behavior mimics that of real-life poker.

The data was initially stored as individual databases. Each database was organized in the relational model, with a table of hands, a table of rosters and a table of player actions for each player. Getting the data into the form we wanted was difficult and the pipeline we used for parsing our data is depicted in Figure 2:

There were a number of different types of hand databases stored overall. We chose to learn on no limit, hold'em tournament data.

4 Models

We decided to try a few out-of-the-box learning models in order to see which algorithms have the



(a) Actions vs. HR (b) Bet Size vs. HR
Figure 3: Feature Correlations

best results on our dataset. Note that we are using the feature set defined above.

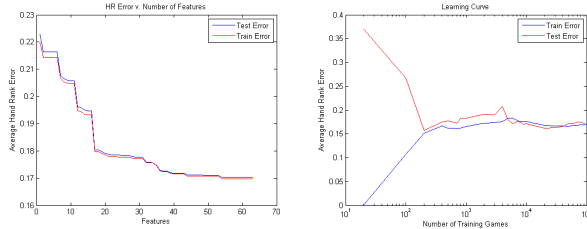
Linear Regression

To motivate linear regression, we first checked for a correlation between our chosen features and hand rank. As shown in Figures 3(a) and 3(b), each feature is indeed correlated with hand rank. Four of the five features: bet size, amount of bets, amount of checks, and amount of calls are positively correlated with hand rank and the amount of checks is negatively correlated with the hand rank.

With the evidence for feature correlation in hand, linear regression was a natural algorithmic choice. To perform linear regression we first expanded our feature set by incorporating not only a player's total game actions, but their individual round actions. The thinking here is that a check during the flop may be weighted differently than a check during the river when trying to predict the river hand rank. Furthermore, we added more features by fitting each action and amount with a fifth order polynomial. To determine the most important features of our algorithm, we conducted a forward search through the feature set. Shown in Figure 4(a), as more and more features were added, the error of the predicted hand rank dropped.

The most important features were found to be: The amount bet during the river, turn, post flop, and pre flop. Next, to determine if we were overfitting or underfitting the data, we generated a learning curve, shown in Figure 4(b).

Since the training error and test error converge in the learning curve, we concluded that we were



(a) Error vs. Features (b) Learning Curve
Figure 4

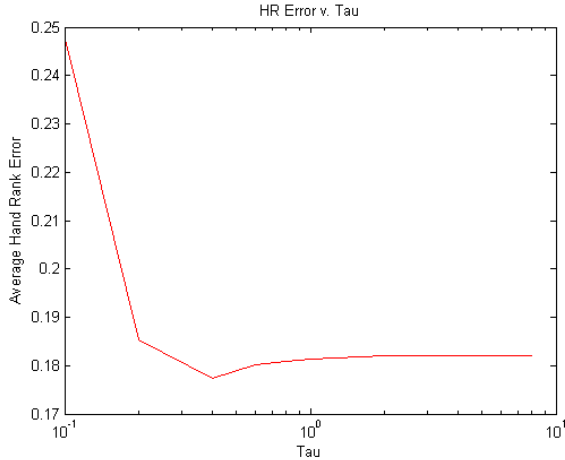


Figure 5: Error vs. τ

underfitting the data. Unfortunately, due to a shortage of any more features, we could not minimize our error any further. We calculated our final average predicted hand rank error on a set of 100,000 games where we trained on 80% and tested on 20%. The average hand rank error was 16.97%. This result was roughly unchanged when linear regression was run on the clustered dataset.

Locally Weighted Regression

Another approach we took to predict player hand rank was locally weighted regression. We used the same feature set with fifth order polynomials as used in linear regression. Since locally weighted regression is more computationally expensive, we only examined 1000 games, training on 80% and testing on 20%. Shown in Figure 5, we minimized the hand rank error by varying the bandwidth parameter τ .

The error-minimizing τ was found to be 0.2. The minimum hand rank error was found to be 17.73%, about 1% greater than that generated by linear regression. This result was also roughly

unchanged when locally weighted regression was run on the clustered dataset.

Naive Bayes

Our goal with our Naive Bayes analysis was to predict hand rank given classes of hands that a player might have at the river of a poker round. We reasoned that:

Players will not change their betting patterns based on exact hand rank. Instead, they will usually change their betting patterns based on general types of hands, i.e. bluffing hand, medium strength hand, made hand (straights, flushes, full houses, etc), or the “nuts (best hand possible).

Certain bet sizes are more likely to indicate certain types of hands earlier in the poker round. For example, we reasoned medium sized bets might be more indicative of a decent or very good hand, while players may make more small bets when they are waiting for a card to complete their hand.

Types of actions on each round may indicate different hands. For example, someone who is calling in earlier rounds but raising in later rounds may mean they had a weak hand at first but a strong hand after a certain card.

We decided on using six bins. We wanted enough bins to make the information useful and applicable, but not so many that players would have the same behavior over different bins.

Choosing Bins and Features for Naive Bayes

Because Naive Bayes is a classification algorithm, we had to transform our continuous hand rank and bet sizing features into discrete bins. We found that having different bins for these greatly changed the results, so we strived to have a bin distribution that was both logical and produced good results. We started the models with even bin distributions for both hand rank and bet sizing, but in the end we decided to primarily change the hand rank bins due to the nature of hands and hand rank distribution on hands that get to the river. See Figure 6 for our final hand rank bin distribution.

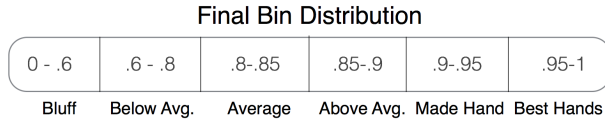


Figure 6: Final Bin Distribution

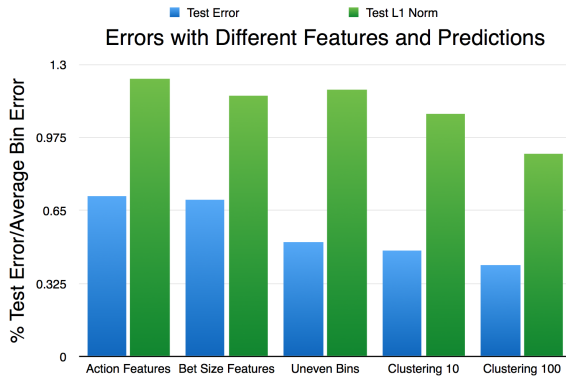


Figure 7: Error Rate Improvement

Results for Naive Bayes

When we first ran our Naive Bayes analysis with a simplified feature set, the results were underwhelming on the human data set and mediocre on the poker bot data. As we added in bet sizing features and more sensible hand rank bins, our results greatly improved to almost perfect on the bot data and about 50% exact bin prediction on human data. Lastly, predictions using clustered data input improved the human data prediction rate to over 60% with a .903 average bin error. Figure 7 shows our results as we iterated on our model.

5 Clustering Player Types

The next idea we had was to model different types of players differently. Anecdotally, there are a number of different types of poker players. For example, there are aggressive players who bet and raise often, and there are also more risk-averse players that fold readily when they don't have a winning hand. We would expect a risk-averse player and an aggressive player to bet very differently given the same hand, but there is nothing in our model that captures this. For this reason we decided to use K-Means clustering on all player data in order to model different types of players

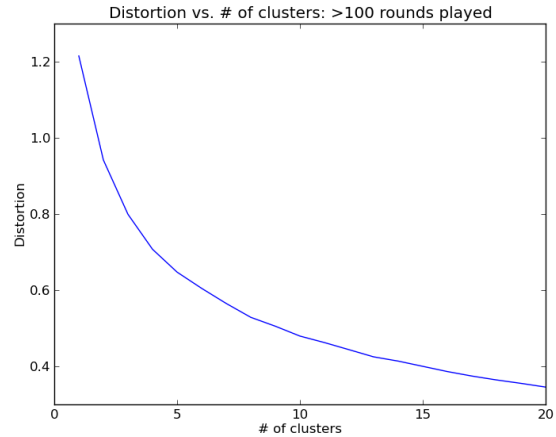


Figure 8: Distortion vs. k

differently. Unlike our Naive Bayes and Linear Regression, we could use all the data we had to determine how aggressive a player is. It was no longer necessary that we actually saw the player's hand at the end of the round. For clustering we used two features.

Features:

- $AggressionFrequency = \frac{Bets+Raises+Allins}{AllActions}$
- $AvgRoundsPlayed = \text{Avg. \# of rounds before the user folds}$

Our idea was to cluster the data into k clusters and then partition our original dataset so that we train on each type of player instead of the entire dataset. The question then became, how many types of players are there, or more formally, what is the best value of k ? Figure 8 plots the average within cluster variance (distortion) vs. the number of clusters.

From Figure 8, we can tell that there is not strong clustering in the data and also no clear choice for k . We tried 3, 4 and 5 and found that 3 clusters worked the best when we partitioned the data and retrained our models. After running k -means (with $k = 3$), our final clusters are depicted in Figures 9 and 10. These diagrams are dependent on what threshold we placed on the number of player actions we had to see to cluster them.

Partitioning the data and training on different models helped our Naive Bayes model jump approximately 10% in accuracy.

6 Conclusion

In conclusion we used a player's actions and betting patterns to predict their hand rank. Using three machine learning algorithms: linear and weighted regression, Naive Bayes, and clustering, we were able to improve our hand rank predictions quite drastically.

Our final Naive Bayes algorithm on clustered data was able to predict the exact bin (out of 6) that a poker player's hand was in 60% of the time and on average the prediction was off by less than one bin. At first, we thought that predicting human trends would be extremely hard due to the nature of poker (with humans trying to trick each other), but we ended up with an algorithm that could perform well without extensive data on a specific player.

One issue we had in our human data was that it was incomplete due to the nature of our data source. We hope that our algorithms could perform better with complete information of every hand instead of only certain hands, which could be explored in the future.

7 References

- [1] Tefilo, L. F., Reis, L. P. 2011. Identifying Players Strategies in No Limit Texas Holdem Poker through the Analysis of Individual Moves.
- [2] Computer Poker Research Group. University of Alberta, 2013. Web. 5 Dec. 2013. <<http://poker.cs.ualberta.ca/>>.
- [3] Poker Academy Pro -The Ultimate Poker Software. 2013. <<http://www.poker-academy.com>>
- [4] Open Meerkat Poker Testbed. 2013. <<http://code.google.com/p/opentestbed>>

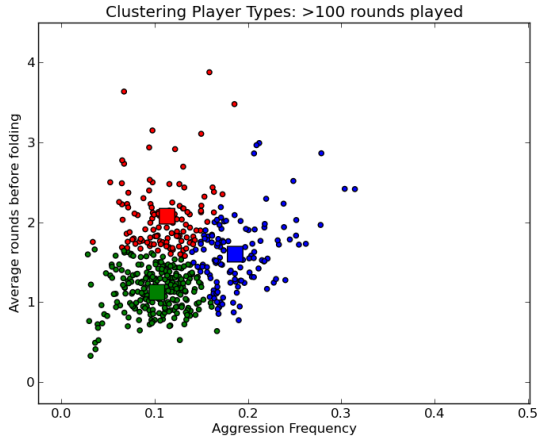


Figure 9: Clusters, > 100 Actions Seen

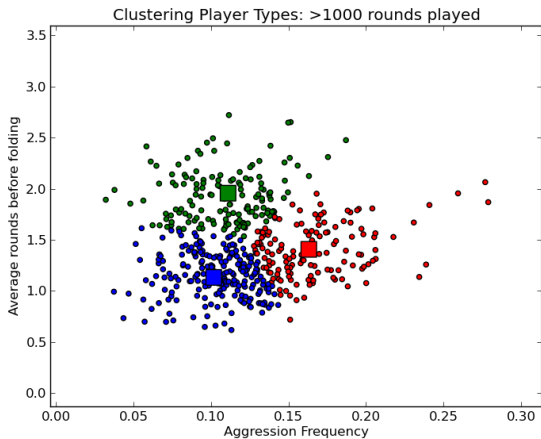


Figure 10: Clusters, > 1000 Actions Seen