Effectiveness of the High Card Point System in Bridge

Aaron Igra

May 24, 2024

1 Introduction

Bridge is a trick taking game played in teams of two. During the course of the game, players communicate information about their hands with goal of predicting how many tricks they will be able to take. Players typically rely on heuristics to do this task. These heuristics are incredibly important in bridge. This is for two reasons. First, humans need a way to quickly and effectively evaluate a set of cards they will never see again. Second, players are only able to pass highly limited information to their partner about their hand through the bidding. In this project, I evaluated a well known heuristic in bridge called high card points, and proposed modifications that make the method more effective.

In order to perform this task, I generated over 250,000 bridge hands, and used a computer algorithm to approximate perfect human play. I compared a traditional heuristic that uses specified points values pertaining to the rank of cards against supervised learning models to predict if a certain number of tricks can be taken when the game is played.

Ultimately, I found the traditional high card point model provides similar error rates as learned models, however the learned models tended to balance the false positive and negative rate.

2 What is Bridge

The game of bridge is played in two phases, a bidding phase, and a playing phase. 13 cards are dealt to each player, and players take turns bidding to find a contract. A contract consists of a suit[1] along with a promise of how many tricks they will take. An example contract is 2♠. This contract promises that during the playing phase, the team will take eight tricks, as a contract always refers to six plus the number bid. During the bidding, each player takes turns

[1] Bridge contracts can be any of the of five options: ♣, ♢, ♡, ♠, and NT, where NT stands for no trump suit.

either bidding a new contract which is higher[2] than the previous or passing, meaning they accept the current contract. Once three players pass in a row, the most recent bid is the final contract.

In the playing phase, there are 13 tricks. In each trick, a player begins by leading a card. Each other player must then play a card of the same suit if they have one, or any card remaining in their hand if they no longer have a card of the lead suit. Whoever plays the highest card of the lead suit wins the trick, unless someone played a card in the trump suit, in which case the highest card of the trump suit wins. At the end, one partnership will have won x tricks, and the other partnership will have won $13 - x$ tricks. If the team with the contract makes the number they promised, they score positive points.[3]

3 Data Generation

Evaluating optimal bridge play is a very difficult problem, as unlike perfect information games such as Chess and Go, bridge has large amounts of hidden information. However, there exists an algorithm for solving bridge deals known as Double Dummy Solver (DDS). DDS assumes all players have perfect information and play optimally. Given this assumption, the DDS can evaluate the optimal decision tree for all players using an algorithm such as mini-max algorithm. This gives us a total number of tricks taken for a given deal, declarer,[4] and trump suit. These results are not equivalent to perfect human play, as human players do not have access to perfect information. Theoretically, this divide could render the generated data significantly different from human play, but in my experience the DDS result tends to align closely with optimal play.

In order to generate the hands, I plan on using a few different resources. The first is the endplay library, a library that can be used for generating and analyzing bridge hands. Importantly, endplay also has a wrapper for Bo Haglund's Double Dummy Solver, which is an optimized version of DDS.

Using google colab, I was able to generate and solve slightly under 10,000 hands an hour using the default CPU and compute resources. Over the course a few long running sessions, I generated 253,583 hands. However, in my analysis I evaluated the north south and the east west pair separately, which doubles the size of the training data.

The final dataset can be found in the github repository. The cards table represents which hand each of the 52 cards is in. The contracts table gives the double dummy result for each suit and declarer pair.

[2] Any contract with a larger number is greater than any bid with a lower number, for two bids with the same number, the contracts are ordered lowest to highest ♣, ♢, ♡, ♠, NT.

[3] The exact amount a contract is worth is incredibly complicated. Check out Bridge Scoring for specifics.

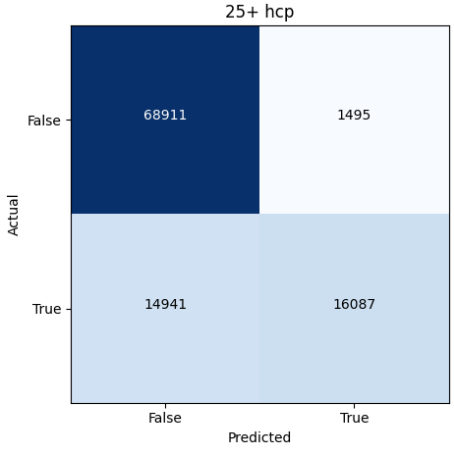
[4] The declarer is the player in the partnership who first bid the suit of their final contract. The player to the left of the declarer plays the first card.

4 Optimizing High Card Points

The first hypothesis I am interested in testing is high card points (HCP). HCP are a well known heuristic for evaluating the strength of a hand. The formula assigns a weight to each face card in the hand. Aces are worth four points, kings three, queens two, and jacks one. All other cards are worth zero. The sum of these cards determines the number of HCP in the hand. A further heuristic I've been taught is that if the sum of you and your partner's high card exceeds 25, you should bid to a game[5].

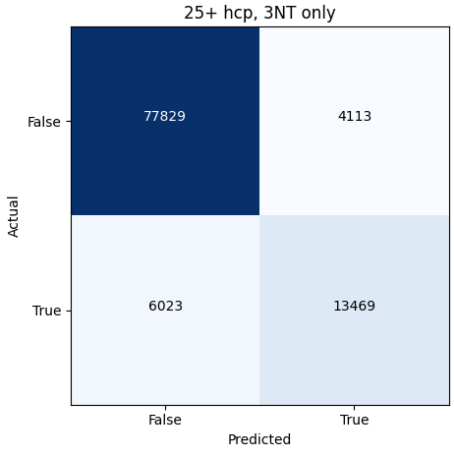
In the following evaluation of high card points, I use a test sample of 20% of the generated hands, which is 101434. This is the same test set that the learned high card point values are tested against later.

It turns out, using the high card point method as written returns a very high false negative rate, with nearly as many false negatives as true positives in the sample.



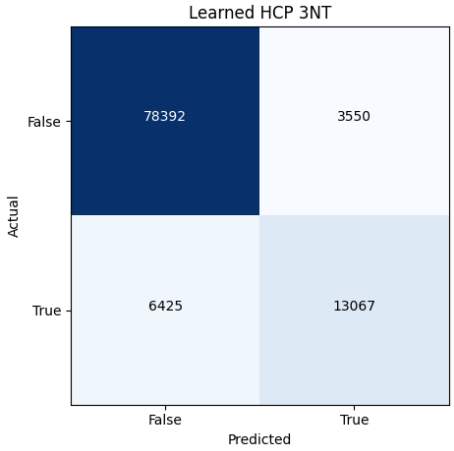
However, high card points are generally only part of the equation if a game makes. Suited games often rely on using the trump suit to overtake high cards of other suits, but may require fewer high cards outside of the trump suit.

[5] In bridge, game contracts are contracts that award a large number of bonus points if you bid to or above them. They consist of 3NT, 4♡, 4♠, 5♣, and 5♢.



By isolating for only 3NT games, we find that the 25 hcp method is significantly more accurate, as it generates only a 10% error rate as opposed to a 16% error rate for predicting any game. Notably, the true and false positive rate is far more balanced.

However, instead of using fixed values to determine the worth of aces, kings, queens and jacks, we can use logistic regression to estimate these values. In theory, the estimated values should have higher accuracy than the original heuristic, as the original high card point values are within the domain for the regression. Upon applying logistic regression to the same features, we can obtain optimal weights for the same features. Training to predict whether 3NT makes, we gain the weights $[3.22, 2.11, 1.22, 0.61]$ for each of the face cards, along with an intercept of -17.88. To ease comparison, we normalize the weights such that their sum is equal 10, which is the same as the sum of the original weights. This normalization does affect the loss of each sample, but does not affect the accuracy, as each sample times the weights will still remain on the same side of the new hyperplane. The normalization step gives us the weights $[4.49, 2.94, 1.70, 0.86]$ with an intercept of -24.94. This means if we use the weights above, the hands must have 24.94 points or higher to be predicted to make 3NT more likely than not. Surprisingly, the values computed are very similar to the original heuristic. Beginning with the intercept, the scaling method resolved with an intercept of 24.94, which is a less than 1% difference relative to the 25. Additionally, Kings remain nearly the same weight, with a difference of less than 2%. However, if we compare to kings, aces were significantly undervalued by the traditional formula, with their real weight being half a point higher, which means aces are 50% more valuable than kings as opposed to only 33% more valuable in the traditional system. This is balanced by the decrease in values for queens and jacks, which both fall by around 15% of their original value.



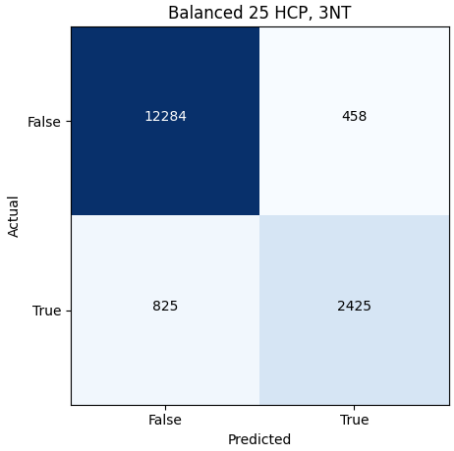
Despite all these changes, the confusion matrix for using the learned high card point values does not show significant improvement over the original. While the false positive rate fell from 4.0% to 3.5%, the false negative rate increased from 5.9% to 6.3%. This means there are only 0.1% fewer errors in the new schema. Overall, while these results do highlight the importance of Aces, the learned schema is not much better than the original heuristic.

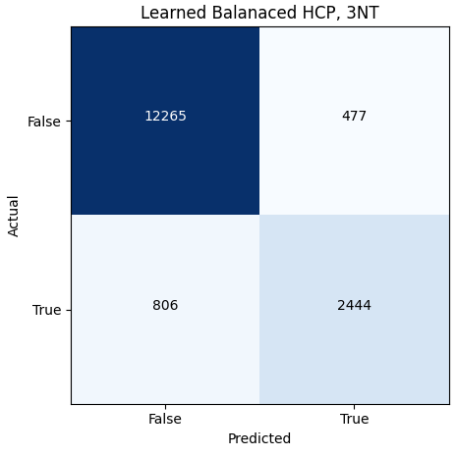
5 A smaller sample set

One theory I had for why the high card point method originally had a high false negative rate was due to the presence of highly distributional hands which relied on a large number cards in a particular suit to make a suited game when no trump wouldn't make. Because our features didn't take into account the number of cards in an individual suit, it is very difficult to predict whether suited games make. Because I still wanted to investigate the high card point method, I chose to filter out hands that have a suit fit. A suit fit is defined by 8 or more cards of a specific suit between an individual and their partner. By eliminating hands with suit fits from our sample, almost all hands will make at least 3NT if they make any game, as they cannot rely on having a large number of cards in a trump suit.

Unfortunately, the majority of hands do have a suit fit. By eliminating those hands, our sample size is reduced to 16% of the original dataset, or 79,956 hands. Using an 80%-20% test split, that leaves our test set at only 15,992 hands.

On the test set, the original high card point method performs well, and the results for the learned method are nearly identical, with the number are false positives increasing by the same amount as the number of false negatives.





Of note, when inspecting the learned weights, they are nearly identical to the original heuristic. We get the scaled values $[4.05, 2.93, 1.92, 1.09]$, with an intercept of -24.31. This system lowers the threshold by a small margin, but keeps the weights practically identical.

6 Conclusion

Overall, I found the HCP system to be far closer to an optimal heuristic than I anticipated for determining whether 3NT makes on any given hand. While it undervalues aces in the average case, on perfectly balanced hands the system is nearly optimal, and learned weights aren't significantly different. This is surprising to me as I suspected the impact of a learned weights would create better accuracy, and would reduce the rate of both false positive and false negatives. However, in all cases the learned weights reduced one type of error but increased the other by a similar amount.

Another filtering tool I tried that was not discussed was looking at hands that were semi-balanced (both individuals have at least two cards in every suit), but this filter had very similar results to looking at hands that didn't have fits.

I also looked at the value of lower rank cards like 9's and 10's. 10's did have a positive impact on making games, with logistic regression valuing them with a weight of 0.3. However, this didn't make much of an impact on accuracy.

Overall, I'd like to do a lot more work on this dataset and like ones that may have more data. In the future, I'd like to evaluate how to select the game that is most likely to make using the number of cards of each suit as a feature in addition to high card ranks. Another avenue I want to explore is how much better neural networks are at predicting games than simple rule based methods. It may also be possible to extract useful information about bidding from the neural network.