CS112 Final Project - Statistical Inference using Hearthstone Data

Yoav Rabinovich

Hearthstone is a popular competitive video game by Blizzard Entertainment, with over 70 million active players as of 2017. It's a competitive card game in which players strategically compile a 30-card deck from a selection of hundreds of unique cards, and are battle one another in short matches. The game requires strategy both in battle and in the deck-building stage, when one must consider constraints, synergies, probabilities of drawing certain cards and importantly the popular decks against which they can expect to face off.

In 2016, Dr. Elie Bursztein published a paper in which he used full replays of Hearthstone games to produce a tool that predicts an opponent's next move based on the cards they played so far. He also included a section predicting the outcome (win or loss) of any given game using statistics about the game state at a given point in the match. Bursztein was asked by the developers to stop supporting the application, since it had a significant impact on player performance that they deemed unfair. Nevertheless, I think his analysis still leaves room for more insights that can be valuable to players.

My attempts to retrieve the original data or analogous replay data didn't bear fruit, but instead I have built a tool to extract data from a popular hearthstone site, HearthPwn, in which players post deck compositions, and extracted 20,000 decks in a csv format. This data lacks information about actions taken ingame, or outcomes of matches. However, with the simplifying assumption that the decks posted on the site are similar to the decks seeing play (both because they originate in the game and because many players use the decks from the site), and the assumption that the community rating of a deck is correlated with its rate of winning, we can produce goals analogous to Bursztein's: Can we predict the composition of an opponent's deck using the cards they have already played? And can we predict the rating of a deck based on the cards it includes?

Deck Classification

The first problem we tackle is the prediction of a 30-card deck from a subset of cards observed from it. We could use logistic regression for every possible card, gauging the contribution of any other card to the probability of its inclusion in a deck. We would then predict chances using all our regression models to pick using our observed cards, and pick the list of cards with the highest probability of inclusion. Since we have over 4,000 cards, that would require 4,000 regression models each with 4,000 features, which is very computationally expensive, and will not be very accurate for most cards since our training data has only 20,000 observations.

We can instead use logistic regression to use cards to predict pre-defined deck archetypes. HearthPwn lets posters tag their own decks with an archetype, and so we can limit ourselves to building a model for each archetype. I split the data into training and test sets, and train a linear regressor for each of the 74 listed archetypes. Aggregating all regression models and choosing the one that is most certain of a positive prediction for all cards in a deck, a precision of 74% and recall of 59% was obtained. This is not bad considering that out of the 74 options the false positives and true negatives are likely to confuse between similar decks. In real use, the model will have to predict an archetype using observations of less cards, but examining the prediction with increasing sample sizes, Hearthstone is sufficiently predictable to reach a precision of 65% and recall of 44% even with a mere 5 cards.

To improve the model, we can kill two birds with one stone. First, our output is only in the form of a single 30-card list representing the most common deck of one of 74 archetypes, which is less informative than if we had more accurate categorization. Secondly, our dataset is flawed: 28% of all decks have no archetype listed, and had to have been discarded in preprocessing. The solution to this, is to build our own classifier that will generate its own idea about archetypes, giving us a more informative output while also allowing us to classify our mystery decks. We can do this using unsupervised classification.

We use the K-means clustering algorithm to build our model, and feed it unlabeled data (including the previously mysterious decks) containing just the categorical variables for cards in each deck and an amount K. The algorithm constructs K classes of decks based on similarity in their card composition. I used k-fold cross validation to choose K according to the percentage of variance explained (PVE) metric, since there is no prediction error by which to evaluate our classification. Using the "elbow method", I found that a K value of around 150 archetypes was the most effective model. With the resulting 150 classes, I created new regression models, and evaluated them. The precision rose to 85%, and the recall to 78%, with which I was satisfied.

In conclusion: Bursztein's model was able to predict not only deck composition, but the next card to be payed with a maximum precision of >90%. However, with only deck data, my model provides an output twice more informative than HearthPwn deck archetypes (150 custom archetypes rather than 74), with high accuracy.

Evaluating Card Impact on Rating with Matching

While the first problem dealt with helping a player evaluate his opponent's capability mid-match, we can also use rating data to inform deck building. Since this encompasses a wide array of possible insights, I give a few proof of concept for applications of statistical inference tool to Hearthstone deck building.

Players of the same deck archetypes often argue about the inclusion of even a single card. For example, the archetype "Tempo Rogue" has a variant that includes the card "Prince Keleseth" which was a recent addition by some players, while others disagree with that choice. Defining the inclusion of "Prince Keleseth" as a treatment applied to "Tempo Rogue", we can evaluate the treatment effect on a deck's rating. A simple approach would be to run a regression and look for correlation between the treatment and the outcome. However, the inclusion of the prince is also correlated with other covariates that might be the real source for the influence on the rating. For example, not only are you taking out a card to fit the

prince in, the prince also requires you not to include certain other cards in your deck. But it's not only deck composition that needs to be examined: as mentioned above, Keleseth was a recent addition to Tempo Rogue, it could be a shift in the meta-game (a term referring to the popular decks in the community at a given time) that would have favored any Tempo Rogue deck, and so it's the correlation between the inclusion and time that disguises itself as treatment effect.

To deal with this without resorting to parametric assumptions, we use matching to find balance in our datasets. We first run a genetic matching algorithm, that allows us to find the most optimal weight to assign to each covariate to use in our matching, including propensity score (representing the correlation between covariates and treatment). Then we run the matching algorithm: for every treated deck, it looks for a control deck with similar covariate values, and prunes away the rest of the control group. In our case, exact matching cannot work, since the inclusion of Keleseth always means exclusion of at least one card, but we find several good matches to each treated deck, and improve our balance from a minimum p-value of 0.03 to 0.31 in the process.

Now we can run a regression on our matched data, and estimate a treatment affect. It seems that the inclusion of Prince Keleseth has an average impact of 30 points in rating, with a 90% confidence interval between 9 and 42, which is users will agree is very large.

Conclusion

Harnessing the predictive nature of Hearthstone can be a very strong tool to influence strategy both mid-match and in deck building. With proper replay data, which I will receive hopefully very soon, I will be able to improve my own playing, and build tools to help others to the same.