Milestones 4/5

- Robert Shaw, Sean Coleman, Spencer Evans, Daniel Alpert
- Data Driven March Madness

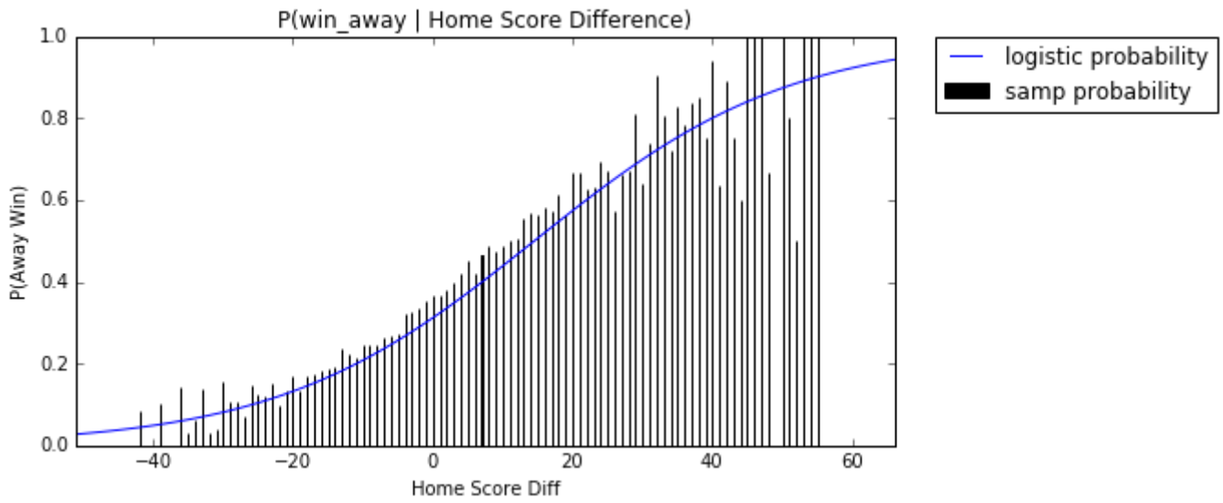
1) Discussion

1.1) Markov Chain Analysis

Methodology Taken From: http://www2.isye.gatech.edu/~jsokol/ncaa.pdf (http://www2.isye.gatech.edu/~jsokol/ncaa.pdf) (see Bibliography Below)

Consider the stationary distribution of a Markov Chain. Each of the π_i represents the proportion of time spent in a given state in the long run. In our Markov Chain, each state represents a given team in the NCAA. At a given time, whichever state the Markov Chain is in is thought to be the "best" team in the country. During a given transition, the Markov Chain reevaluates its "best" team, and transitions to another state with some probability p_{ij}. We create the p_{ij}, by asking the question, "Given that team i has a score differential of x against team j, what is the probability that team i is better than team j?. We define this probability to be r_x^h.

To answer this question, we use structure of the NCAA basketball season. Becuase of the "home and home" structure of the in-conference games, we can fit a logistic regression model on the probability of a team winning on the away game of a home and home series based on the score differential of the game on the home court. We define this probability to be s_x^h.

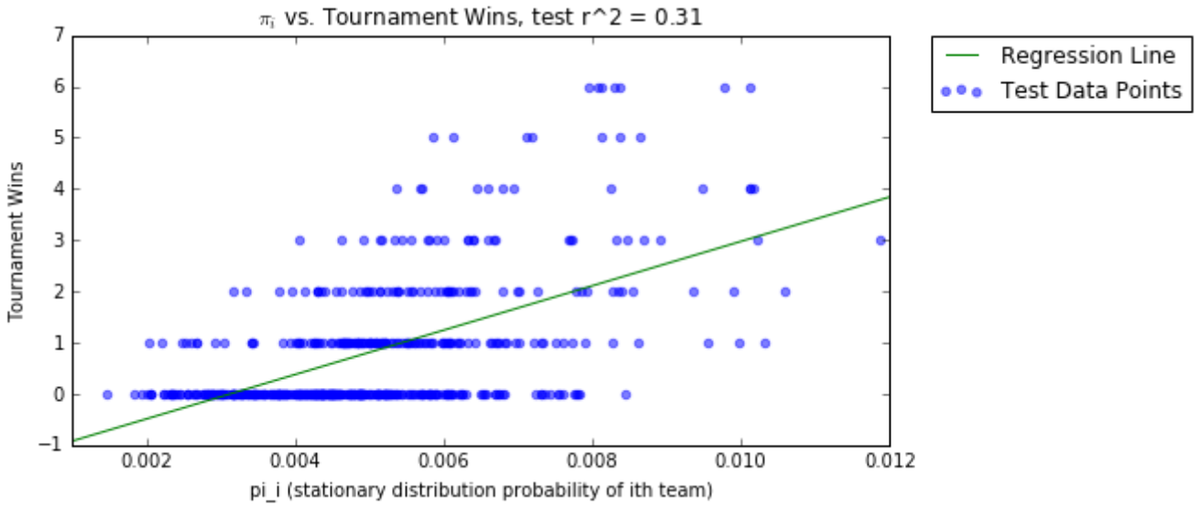


The black bars represent the observed probability of the team winning on the away court given the scoring differential of x on the home court. The smooth blue line represents the probability of the team winning on the away court calculated under the logistic regression model. As we would expect, the higher the score differential in the home game, the greater probability of winning on the away court.

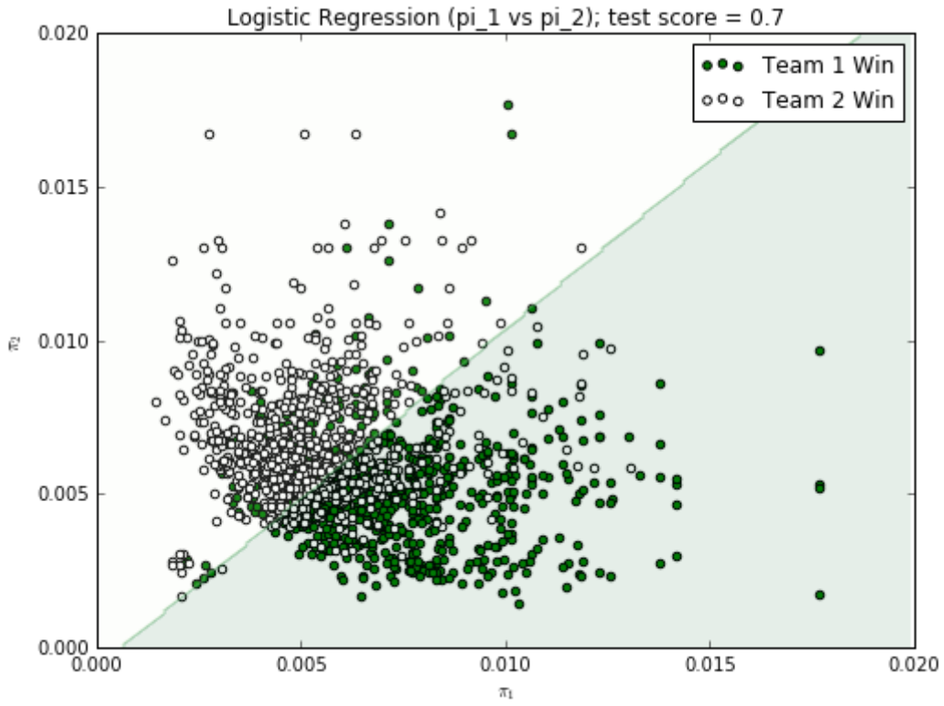
Now that we have calculated s_x^h, the probability of winning on the away court given a score differential of x on the home court, we can find r_x^h, the probability that team i is better than team j, given the scoring differential of x on the home game. To do this, we find the value of x, such that $s_x^h = .5$. In this situation, the expected score on the away court is 0. Thus, this $x = 2h$, since there is 2 times the home court advantage swing between the games. We found this x to be 14, so $h = 7$. Thus, we have $r_x^h = s_{x+h}^h$. We also have r_x^a (the probability of being better given a score differential of x on the road) to be $r_x^a = 1 - r_{-x}^h$.

Now given these probabilites r_x^h, r_x^a, we instantiate the Markov Chain by passing giving edges between teams with weights proportional to r_x^h and r_x^a, depending on the results of the games in the regular season. Since each state has a self loop and all states commmunicate with all other states, this Markov Chain is ergodic, which implies that the stationary distribution exists. Thus, to find π such that $\pi = \pi P$, we find the first eigenvector of P, where P is the transition matrix. This π has some interesting properties. Consider team i. If team i has beaten a lot of teams by large margin, the markov chain will spend a lot of time in state i, since there will be many edges with heavy weights going into i. Now consider team j, which beat team i. Since j beats i, there will be an edge with heavy weight between i and j, meaning that there will be a lot of transitions from state i to j. Since a lot of time was already spent in state i, a lot of time will also be spent in state j. Thus, π_i is related not only to the number and dominance of the wins of a team, but also to the quality of the teams beaten.

To back up this theoretical argument, we implemented this Markov Chain model and tested its predictive power. First, in predicting the number of games a team will win in the tournament, we fit the following linear regression:



As we can see, there is a positive trend between π and the number of tournament wins. After seeing this result, we tested how well π does for predicting the winners of head to head tournament games. We fit a logistic regression model, and found the following relationship. With this, we got 70% accuracy on the test set, showing that just this predictor does well both quanitatively and qualitatively:



Bibliography

Kvam, P., & Sokol, J. S. (2006). A logistic regression/Markov chain model for NCAA basketball. Naval Research Logistics, 53(8), 788-803. doi:10.1002/nav.20170

1.2) Infrastructure

Beyond this Markov Chain Analysis, which will serve as our baseline model for predicting head to head games, we recognise that predicting a bracket is more complex than predicting head to head games. Thus, we setup some classes, which when fed with the appropriate data (including seeds for a given year, raw tournament game data, and the bracket structure) allows us to simulate a tournament under a particular model for predicting head to head games, which can be passed in as an argument. We also provide a mechanism for scoring the model, by passing in the actual results of the tournament and comparing our predicted results to the actual results. This is very important to our model, and was non-trival to implement, so we felt it necessary to include this here. Please request the code if you are interested in seeing how this works.

1.3) Scoring Methodology

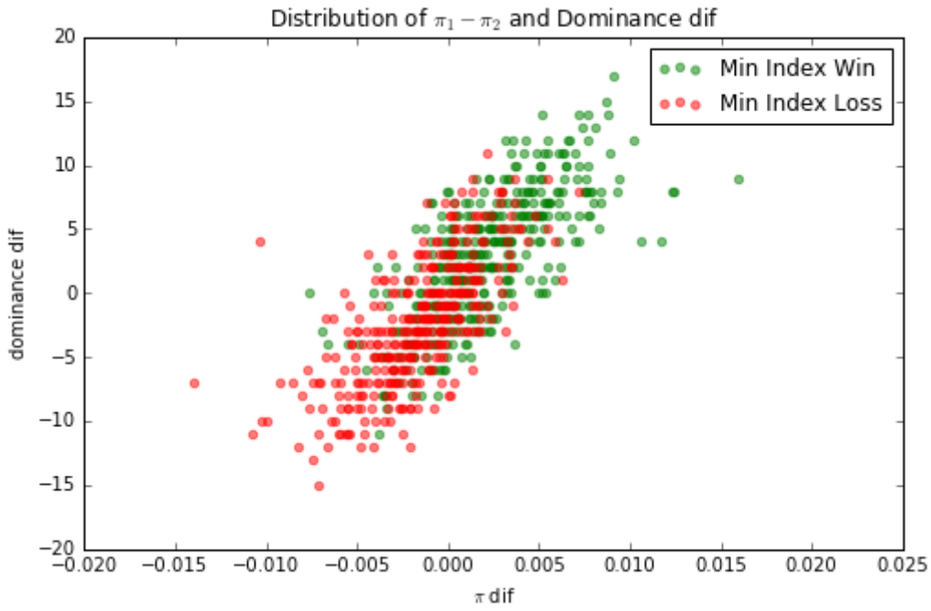
To score the model, we want to emphasize that predicting late round games is more important that first round games. To do this, as most sites do, we will use a points system. ESPN standard scoring gives 10 points for a round 1 win, 20 points for a round 2 win, 40 points for a round 3 win and so forth, such that each round is worth 320 points. We have also allowed our functions to be few different scoring systems as well, such that we can use Yahoo! scoring and others in order to evaluate our model against other models, expert analysis, and the baseline. Beyond just calculating score, however, we also allow our model to see overall accuracy as a percentage to get an alternate metric.

1.4) Baseline Model

Obviously, we will want our model to do better than predicting 50% of the games, becuase the structure of the tournament makes it easy to pick the 1 seed to beat the 16, for example. Thus, we will always compare our model to picking the higher seed in each game. In other words, we will evaluate our model relative to the NCAA tournament commitee's ranking.

Now, given this, we fit our baseline model with 2 predictors, the stationary distributions π and the number of quality wins (wins vs. tournament teams). Obviously, since we have values for each team, each of the predictors x_i are the difference between the values of team 1 and team 2. In order that we have approximately equal populations in each group, we take as the response the probability that the team which has a smaller "Team Id" wins a game.

When we visualized this data, we found the following:



As we can see, the data are multivariate normal with constant covariance matricies, suggesting that the assmuptions of LDA hold. After training both a logistic model and an LDA model, we found the LDA model has a 72% accuracy in predicting head-to-head games in the test set, while a logisitic model has a 69% accuracy.

When we tried predicting an entire tournament with the LDA model with only these 2 predictors, we were pleasantly suprised by its accuracy. We ran the model over years 1985-2000 and found that our model accurately predicted 65% of games, getting an average score of 920.0. For comparison, the "Baseline Model" of predicting the higher seed in each game accurately predicted 65% of games, but only had an average score of 890. As such, it seems that our model is doing well in later round games, where upsets are more likely to happen as there are more evenly matched games. Thus, our baseline appears to be working decently well.

1.5) Next Steps

- 1) Adding More Predictors: Given that our baseline model does a decent job, we want to improve the quality of the model by adding more predictors. Specifically, we are interested in investigating quality wins, consistency, experience, program pedigree, and other "qualitative variables" which we can quantify in some way. We feel strongly that we should only add predictors to the model if they make sense logically, so as to avoid overfitting to the training set. Thus, we plan to generate more predictor variables and to do some more data exploration to add more variables to the model. Further, we will consider switching the model to LogReg in order to have a better interpretation (by looking at the coefficients) to see which variables we ought to include and which we ought not to include.
- 2) A Better Decision Process: Due to our scoring structure, in which late round games are worth more, we may want to alter our tournament prediction to allow teams that are more likely to do well in late rounds to advance. Since late round games are worth more points, sending teams which are likely to succeed in late rounds will give us more points than doing the optimal action in an earier round. We have thought of 2 approaches to this method.
  - First, we could forumulate the problem as an integer programming problem and set the objective function as maximizing the expected number of points. We would precompute all of the probabilties by running a head to head model between all possible teams and use dynamic programming to find the probabilities that a team reaches any round. Then, using binary variables X_{ij}, we could create a set of contraints that only allow a set number teams to move onto any given round. We are concerned that this IP will be too big, and due to the NP-Completeness of Integer Programming, there is not obvious polynomial time algorithm to solve IPs. Thus, we are skeptical of this approach.
  - Second, we could run a simulation of the tournament, taking the head to head results of the head to head model as probability that team i beats team j. By running the simulation multiple times, we can get the average number of points a team scores per tournament. Given this, we can then rank the teams based on expected total points, and then predict outcomes of games based on these expected total points. Obviously, if we do this route, we will need to use some form of logistic regression of discriminant analysis, since we need to get a probability for each head to head game.
- 3) Visualizing The Bracket: We need to write some code to easily visualize the outcomes of the tournament. To do this, we need to write a python script to output some HTML formatted as a bracket.