Edge direction prediction of sporting tournament graphs

Suvrat Bhooshan, Josh King, Wayne Lu

December 11, 2016

1 Introduction

In the realm of professional sports, experts' reputations are partially staked on their ability to accurately predict the results of games and to accurately rank teams by their performance. However, expert predictions and rankings are often only based partially on statistical data and are subject to personal bias. By modeling a sports league as a network where nodes represent teams and directed edges represent the superiority of one team over another, we can objectively approach prediction and ranking as a network analysis problem through the application of edge prediction and ranking algorithms. In particular, we note that a sports league network naturally orients itself via status theory, as teams which are strong will be superior to many other teams and thus have more in-edges, whereas teams which are weak will be inferior to many other teams and thus have more out-edges.

In this project, we investigate the application of network analysis algorithms to sports prediction and ranking and compare our performance against that of experts. Our work can be divided into two distinct sections. The first section involves the building of a league model using historical game data from previous seasons and then using that model to predict edges for a more current season. The second section involves the algorithmic generation of a league ranking for a season and evaluating how well it represents the games of that season, and then comparing the algorithmic rankings against expert rankings.

In building this league models, we use two professional leagues: Major League Baseball (MLB) and the National Football League (NFL). The two leagues were chosen because of the wealth of data available about them and because of their schedules. In the MLB regular season, 30 teams each play 162 games and notably play other teams more than once. In contrast, during the NFL regular season, 32 teams each play 16 games and will play another team at most once. Thus, for a given regular season, the relationship of two teams in the MLB is determined by a series of games whereas in the NFL it is determined by only one game.

2 Related work

In *Predicting positive and negative links in online social networks*[1], Leskovec, Huttenlocher, and Kleinberg describe a method for using logistic regression to predict edge signs in directed, signed networks. In particular, beyond regular topological features such as degrees and common neighbors, Leskovec et al. use the participation of pairs of nodes in 16 different types of triads as features for the learning algorithm. In our work, we adapt this usage of machine learning and triad participation for edge direction prediction.

In *Predicting link directions via a recursive subgraph-based ranking*[2], Guo, Yang, and Zhou describe an algorithm to generate an ordering of nodes by recursively generating subgraphs and ordering the nodes of the subgraphs by their degree difference. The ordering of the nodes is based in status theory, which assumes that edges will tend to point from nodes of lower status to nodes of higher status. In our work, we implement this algorithm and compare its performance to two other variants, which is explained further in section 3.2.1.

3 Approach

3.1 Data and modeling

For each league, the relationships between teams is modeled as a weighted, directed graph. We define the win-loss spread of a team i against a team j as the number of times i won against j minus the number of times i lost against j. Then, for the directed edge (i, j) between teams i and j, the weight of (i, j) is calculated by summing the win-loss spread of j against i across multiple seasons, with a discount factor γ applied to older seasons. Thus, a positive weighting on (i, j) implies that j is better than i, whereas a negative weighting implies that i is better than j.

3.2 Algorithms

3.2.1 Recursive ranking

The ranking algorithm described by Guo et al. operates by recursively ordering the nodes of subgraphs. Beginning with the original graph G, the algorithm orders the nodes by the degree difference $d^{\Delta} = d_{\text{in}} - d_{\text{out}}$, using the assumption that higher ranked nodes should have more incoming edges than outgoing edges. Ordering is determined randomly for nodes which have the same d^{Δ}. Then, the nodes are split into two groups of "leaders" and "followers," V_L and V_F, where V_L contains the first $\alpha|V|$ sorted nodes and V_F contains the remainder, α being a parameter of the algorithm. The process is then repeated for the subgraphs induced by V_L and V_F until a global ordering is achieved. We notate this algorithm as "D-R" because it sorts nodes primarily by degree difference, then tiebreaks randomly.

We also implemented two variations of this algorithm. For a node i, we define the edge weight difference as

$$d^w(i) = \sum_{(j,i)\in E} w(j,i) - \sum_{(i,j)\in E} w(i,j)$$

where $w(i,j)$ is the weight of edge (i,j). The first algorithm variant, denoted "E-R", sorts nodes primarily by edge weight difference, then tiebreaks randomly. The second algorithm variant, denoted "D-E", sorts nodes primarily by degree difference, then tiebreaks by edge weight difference.

3.2.2 Others

In addition to the recursive ranking algorithm, we applied the HITS algorithm to the graphs and use the authority score for each node as a ranking. These rankings can be used as a stand-alone method for prediction, or can be combined with other features to create more accurate classifier. We also used a standard implementation of logistic regression for a machine learning approach to edge prediction, as described in the section below.

3.3 Methodology and evaluation

3.3.1 Game prediction

To predict game results for a season, we use two different approaches. The first is a machine learning-driven approach which trains a logistic regression algorithm to predict the direction of an edge between two teams, which in turn represents our prediction of the result of a game between the two teams. The primary features that we use are triads. Similar to Jure et. al we train our classifier using partial triads between all pairs of nodes, and the corresponding label of the edge between those nodes. More formally the features between nodes u and v would be $\{(sign(u,w), sign(w,v))\forall_{w\in V:(u,w)\in E,(w,v)\in E}\}$ with the corresponding label being $sign(v,u)$. We dropped directionality and instead used the order as the directionality. So all triads started at u, went to w, then to v, and then back to u. Because the features consist of only two edges, and each can have two values, this leaves only four features: u beats w w beats v. u beats w, w loses to v. u loses to w, w beats v. u loses to w, w loses to v. So the value of each of the attributes for the edge from u to v would be the count of each of the four types of triads.

Given a machine learning classifier, there still need to be known edges in the graph that we are testing on in order to be able to generate the features for the edges we are trying to predict. More formally, if we are trying to predict the edge from u to v, there must exist some node w, s.t. $u, w \in E$ and $w, v \in E$. So the triad classifier would not be able to make predictions about the first games of the season. Because of this, for all triad classifiers we used k-folds cross validation with 4 folds. While our folds were generated randomly, if they were generated in chronological order this would be analgous to predicting the outcomes of the last quarter of a season when you are three-quarters of the way through the same season. For all of the models in Table 1 we do not use the edge weights, only their sign. We tried using edge weights but that decreased the accuracy.

We created five different methods of using triads, and the results are shown in Table 1. All methods use 4-folds cross-validation to generate accuracy. They use three folds to generate features for the edges in the other fold, and if the model is training and testing on the same graph it uses those three folds to train the model. The Triads method uses only triads as features and trains and tests on the same dataset. The Different Year method trains on the 2010 season of the same sport (except when testing on 2010 which trains on 2011) and tests on the given dataset. Different Sport trains on the 2010 season of the other sport and tests on the given dataset. Triads with HITS is the same as the first column except for with the authority score of each of the two teams added as attributes. Triads with HITS All Data trains on every other dataset except for the test set. So 2010 NFL would train on 2011-2015 NFL and 2010-2015 MLB.

For the machine learning classifier we used Logistic Regression with l2 loss and regularization and scaled all features. We also looked at Gaussian SVMS, Linear Regression (used to predict the weight of the edge and then just take the sign), and Naive Bayes. All performed similarly so for the sake of brevity we only included the results from Logistic Regression in this paper. The second approach is to generate rankings of the teams by running ranking algorithms on historical data. Game predictions can then by made by predicting that a higher ranked team will win against a lower ranked team. For this approach, we use the D-R, E-R, and D-E recursive ranking variants, as well as PageRank and HITS.

For both approaches, the accuracy of an algorithm is determined by the proportion of accurately predicted games. In generating a model of the historical data, we set our discount factor $\gamma = 0, 0.1, \ldots, 1.0$. In addition, for the D-R, E-R, and D-E algorithms, we use $alpha = 0.1, \ldots, 0.9$, and then take the most accurate result.

3.3.2 Ranking

To evaluate our ranking algorithms, we generate rankings for the league graph model of the 2015 season using the D-R, E-R, D-E, and HITS algorithms. A game is correctly represented by a ranking if the winner is ranked higher than the loser. The accuracy of a ranking is then calculated as the proportion of games it correctly represents. For the D-R, E-R, and D-E algorithms, we use $alpha = 0.1, \ldots, 0.9$ and take the highest accuracy result. As a baseline for the rankings, we use the accuracies of the 2015 MLB standings and the Week 18 NFL Power Rankings.

The rankings from HITS are generated using the authority score of each node. To generate this node we only use the edges from u to v where u loses to v. The higher the authority score the higher the rank of the node. We ran HITS for 1000 iterations.

4 Results and findings

4.1 Game prediction

Table 1: Triad Edge Prediction

Dataset/Method	Triads	Different Year	Different Sport	Triads with HITS	Triads with HITS (All Data)
2010 NFL	0.60	0.60	0.64	0.62	0.55
2011 NFL	0.60	0.61	0.63	0.64	0.60
2012 NFL	0.65	0.61	0.61	0.64	0.59
2013 NFL	0.62	0.62	0.66	0.62	0.59
2014 NFL	0.66	0.62	0.66	0.62	0.59
2015 NFL	0.61	0.58	0.66	0.60	0.56
2010 MLB	0.65	0.66	0.65	0.65	0.65
2011 MLB	0.56	0.55	0.56	0.56	0.55
2012 MLB	0.65	0.65	0.65	0.66	0.65
2013 MLB	0.61	0.61	0.62	0.62	0.62
2014 MLB	0.59	0.57	0.58	0.60	0.58
2015 MLB	0.58	0.59	0.58	0.58	0.59

Table 2: Predictive accuracy for various triad edge prediction methods.

The results in Table 1 show various triad edge prediction methods. There are several findings. First, the overall accuracy is comparable to other prediction methods. Second, the HITS score sometimes helps. Thirdly, as shown with other networks the status models are highly transferable. Finally, more data to train on is not necessarily helpful, but more features are.

Our average accuracy on NFL datasets using Triads is 0.63. While this is by no means a very high accuracy it is on-par with other cutting edge statistical models. We could not find a comprehensive review of NFL/MLB predictions so instead we looked at SBNation's picks and Fivethirtyeight's predictions. To make it a fair comparison we only looked at their predictions for the last quarter of each season because our models saw a similar amount of data. Fivethirtyeight only did formal predictions for 2015 and their accuracy for the last quarter of the season was 0.63, slightly better than vanilla triad's accuracy of 0.61, and worse than using triads learned on 2010 MLB data. This is surprising because their model uses many more features, delving into individual player stats. Whereas our model only uses the triad features. SBNation's accuracy was a bit higher, they averaged around 0.70 for all NFL seasons. Again though, they are using many more features than our model.

The HITS score for each team as attributes can be helpful. It does not make a significant difference most of the time, but it never hurts. This suggests that it sometimes is adding information that the triads do not capture. This suggests that other features that try to develop global information about the network are likely to improve accuracy. It in and of itself does not perform very well though. The HITS scores can only predict about 0.55 of the games in and of itself.

The Triads and HITS with All Data which trains on all data besides the given season is no better, and is normally worse than just training on a single other season. This indicates that the limitation is not the machine learning classifier, and is instead the amount of information that triads hold. However, when we decreased the folds in cross-validation from four to two, we saw a dramatic decrease in performance with no dataset getting about 0.60 for accuracy. After four folds though, there is no performance gain. This suggests that you need a fairly connected graph in order to accurately predict edges, but once you reach that ceiling there is only so much that can be gained sheerly from looking at triads.

All of the networks are very similar to each other. When learning on one network from another season or even another sport and testing on another there is no decrease in performance. Sometimes, as in the case of training on the 2010 MLB network and testing on the NFL networks, there can be a signicant increase in performance. This suggests that these networks have very similar characteristics. When looking at the learned coefficients for each model they are very similar. For example, the 2010 NFL model coefficients are [-0.14766841 -0.00421349 0.00563206 0.14628048] and the 2015 NFL model coefficients are [-0.11565131 -0.0037338 -0.00393853 0.12436987]. This is in line with previous work by Jure et. al, although they saw some perforamnce when training and testing on different networks.

League	D-R	E-R	D-E
MLB	0.53	0.53	0.53
NFL	0.64	0.64	0.63

Table 3: Ranking-based game prediction accuracy (2015 season)

4.2 Ranking

League	D-R	E-R	D-E	HITS	Baseline
MLB	0.58	0.59	0.59	.57	0.57
NFL	0.77	0.79	0.77	.69	0.68

Table 4: Ranking algorithm accuracies

We find that our algorithmic rankings have noticeably higher accuracy than the expert rankings with respect to representing the games of a season. Surprisingly, the accuracies for the MLB are lower than the accuracies for the NFL, despite the larger amount of data available for the MLB. There is no significant difference in accuracy between the three variants of the recursive ranking algorithm. Table 5 has an example of the rankings created by HITS, "E-R", and NFL ELO scores for NFL 2015.

E-R	DEN	ARI	CIN	NE	CAR	MIN	...	PHI	NYG	MIA	TEN	BAL
HITS	ARI	WAS	SF	KC	GB	PIT	...	DAL	MIA	BAL	CLE	TEN
ELO	STL	NE	GB	DEN	DAL	IND	...	OAK	TB	WAS	JAC	TEN

Table 5: Rankings for NFL 2015, 1 on the left, 32 on the right

5 Conclusion

It is worth noting that game results are not deterministic and that in general, sports games as a random process are incredibly noisy. In addition, in comparison to other real-world networks such as Facebook or Wikipedia, our league models contain significantly fewer nodes and edges (though the graphs are denser than other networks). Despite these limitations, by generating models of sports leagues using only historical win-loss data, we are able to consistently predict game results with better-than-random accuracy. However, we are normally underperform expert predictions, which is not surprising. Expert predictions incorporate significantly more domain-specific data such as offensive/defensive strategy matchups and individual player statistics. In addition, expert predictions are made on a weekly basis and thus have access to games from previous weeks in the same season. By developing a more complex model which accounts for additional statistical data and updating our prediction methodology to generate a new model for each week of games, it is likely that we can increase our prediction accuracy. In addition, our ranking algorithms are able to generate league rankings which outperform expert rankings with respect to accurately representing game results across a season. This is also not surprising because expert rankings are not made objectively and often rely more heavily on recent games rather than considering the entire season.

6 Works cited

1. Leskovec, Jure, Daniel Huttenlocher, and Jon Kleinberg. "Predicting positive and negative links in online social networks." *Proceedings of the 19th international conference on World wide web. ACM*, 2010.

2. Guo, Fangjian, Zimo Yang, and Tao Zhou. "Predicting link directions via a recursive subgraph-based ranking." *Physica A: Statistical Mechanics and its Applications* 392.16 (2013): 3402-3408.

7 Contributions

Suvrat: Logistic/Linear Regression for Edge Prediction
Josh: Triads models and HITS scores
Wayne: Recursive ranking algorithms, data model generation