Graphs in Machine Learning
Application of Hypernode Graphs Methods to Predict Basketball Games Outcome

Thibault LAUGEL
April 2015

1 Introduction

Predicting games outcome is a major challenge is several areas, such as in the sport and gaming industries, or even in politics when an election approaches. The applications can then vary, from simply predicting the winner of the game to ranking all participants by defining a skill measure.

Multiple approaches can be considered to tackle this problem, by either treating it as a classical Machine Learning prediction problem or by using the games data to build a graph and apply specifics algorithms. However, issues arise when considering this second approach for team games: despite building a graph over teams is easy, it is harder to take into model the interactions between players in each team. The notion of hypergraphs, introduced by Claude Berge in 1969 [1], was an exemple extention of the graph notion to model relations that involve more than two nodes. However, this notion is inefficient to model binary relations between sets of individuals. Hypernode graphs modeling is a way to take into account both granularity levels (teams and players) to study multiplayer games.

In this paper, we focus on applying hypernode graphs methods presented by Riccate, Gilleron and Tommasi [2] to the NCAA Basketball Tournament games from the Kaggle challenge "March Machine Learning Mania 2015"[3].

The first part of this paper will be dedicated to a description of the available data and of the problem. Then, after having explained the methodology, the results will be presented.

[1] Graphs and Hypergraphs, C. Berge (1969)

[2] Hypernode Graphs for Spectral Learning on Binary Relations over Sets, Riccate, Gilleron and Tommasi (2014)

[3] See: https://www.kaggle.com/c/march-machine-learning-mania-2015

2 Challenge and data description

2.1 Data description

The goal of this project is to answer the Kaggle challenge 'March Machine Learning Mania 2015'. Each year, more than 300 college Basketball teams take part in the NCAA Basketball Championship, often called March Madness. Although multiple information were provided for the challenge, applying Hypernode graphs methods only required the teams rosters.

The corresponding data was crawled from the Fow News website, then cleaned and added to the existing datasets. The cleaning process included removing teams with incomplete rosters, matching colleges and teams names and limiting the number of players per team to 5.

2.2 Remaining data and accuracy measure

In order to predict the outcome of the 2015 games, the 2014 results have been used. After cleaning the dataset, this consisted in:

- 348 teams
- 1740 players
- 5 362 games

The following figure illustrates the distribution of the teams depending on the number of games they played in the train and test datasets:

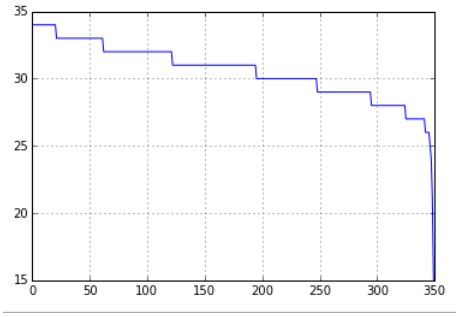

Figure 1: first figure

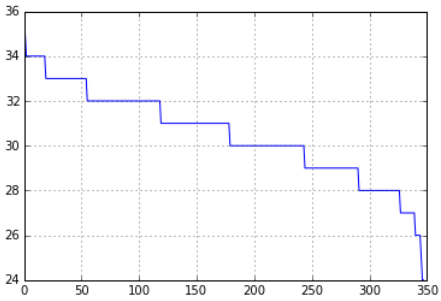

Figure 2: second figure

This distribution shape is due to the tournament format, where teams are moving forward in the bracket depending on their results. In average, a team played 30,5 games.

It is important to note that only 859 out of the 1740 players of the testing dataset are known from the training dataset.

We focus now on the methodology used to predict the games outcome.

3 Methodology

As explained earlier, the methodology used in this project relies on the notion of hypernode graph.

3.1 Hypernode graphs

Considering that our tournament is a set of games $\gamma_j, j \in 1, ..p$ each played by two teams A_j and B_j, each composed of 5 players in our set of players $\{1, ..., n\}$, we can build a hypernode graph by grouping the players of a same team in a $hypernode$ in order to model the binary relation between two teams while taking into account each player. Thus, each team corresponds to a hypernode and each game to a hyper edge. The following figure is a way to represent a hypernode graph.

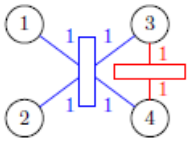

Figure 3: Example of hypernode graph with 2 games

The previous figure represent a hypernode graph with 2 games, one between players $\{1, 2\}$ and $\{3, 4\}$ and the other between player 3 and player 4.

Although a basetball team has different roles that have different impact on a game, since we have no information on the role occupied by the team members we choose to consider that each has the same contribution to the game, set to 1.

In order to predict the game outcomes, we want to determine which team is the best by computing an individual skill value for each player. To do this, we create for each game a virtual node (called $outcome\ node$), part of the losing team, with a skill equal to the difference between the two teams scores. This node is set to have a contribution of 1 to the game. In order to preserve the Equilibrium Condition[4] between the two teams, we also define another virtual node called $lazy\ node$, with a contribution to the game of 1 and a skill of 0. The following figure illustrates the construction of a hyperedge between two teams, with the corresponding virtual nodes:

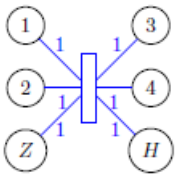

Figure 4: Example of hyperedge with the corresponding virtual nodes

In the previous figure, the game modeled by the hyperedge was a victory for team $\{1, 2\}$. Thus, 2 virtual nodes were created: an outcome node H and a lazy node Z.

Although each game needs a lazy node and a virtual node, it is equivalent to consider that the lazy node is a member of each winning team, and the outcome node a member of each team losing to its corresponding score difference. Since basketball scores difference are usually lower than 100 and that no draw is possible, we created 100 outcome nodes and 1 lazy node in our hypernode graph.

[4] the Equilibrium Condition states that the sums of the square-root of the contributions of each team must be equal.

To recap, we added 100 outcome nodes and 1 lazy node to our existing $n = 1740$ nodes, for a total of $N = 1841$ nodes.

3.2 Laplacian and weight matrices construction

3.2.1 Laplacian

The $Gradient$ matrix of the graph is a way to model the relations between players that play against each other or in the same team. Letting G be the (p, N) gradient matrix of our hyper node graph, we have:
$G[i, j] = 1$ if player j won game i, -1 if he lost, and 0 otherwise. Thus, following the previous definition of virtual nodes, the entire column of the lazy node is filled by -1 (he won all games), and adding up the columns for each line gives 0 (Equilibrium Condition).

Following the algorithm described in 'Hypernode Graphs for Spectral Learning on Binary Relations over Sets', we then computed the $(p + n, N + 1)$ regularized gradient G_μ by adding a regularizer node R, an hyperedge between every player node and the regularizer node R with node weights μ/n. Finally, the regularized Laplacian matrix is the $(N+1, N+1)$ matrix defined by: $L_\mu = G_\mu^T G_\mu$.

3.2.2 Weight matrix

Using the gradient matrix defined earlier, we used the relation: $G^T G = D - W$ to compute W, where D is the diagonal (N,N) matrix where $W[i, i]$ is the node degree of i which corresponds to the number of games the player i played.

3.3 ZGL algorithm

Using the regularized Laplacian matrix and the weight matrix, we use the ZGL algorithm described in 'Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions'[5] to compute a harmonic solution to the skill minimization problem:

$$\min_{r \in N+1} r^T \Delta_\mu r \tag{1}$$

s.t. $r(n + 1) = 0$ and $\forall n + t + 1 \leq j \leq N, r(j) = o_j$

The harmonic solution of this minimization problem is given by:

$$r_u = \Delta_\mu^{-1} W_{ul} r_l \tag{2}$$

Where the index u refers to unlabeled data, namely our n player nodes, and l refers to labeled nodes, namely our $N + 1 - n$ virtual nodes.

3.4 Games outcome prediction

Once the skill values have been computed for each player of the training dataset, the only remaining thing to do was to use them to predict the outcomes of the games of the testing dataset. We saw that more than half of the players of the testing dataset were not in the training dataset; these players were given the node average skill value.
Finally, adding up the skill values predicted gave the total skill value of the team, and the predicted winner of each game was selected by taking the higher value.

[5] by Zhu, Ghahraman and Laffarti, 2003

4 Results

4.1 First attempt

We computed the results for $\mu/n = 30$, which is approximately the average number of games played by each team.

Our first attempt gave an accuracy of approximately 63%. Although this result seems quite low, a few things can be highlighted:

- The absolute skill gap between the two teams in each game is in average smaller (2,1 against 3,5) for the games that were incorrectly predicted, meaning that the teams were closer in terms of skill when the model did a wrong prediction.
- The teams in the testing set with the less known players from the training set are more likely to have wrong predictions, since their team skill is simply 5 times the average node skill.

4.2 Increasing the training dataset

The first attempt was made using only the 2014 season as the training dataset to ensure faster computation. In order to try increasing the quality of our model, we tried to add previous seasons in the dataset. Thus, two new attempts were made, using :

- seasons 2013 and 2014 as training set
- seasons 2012 to 2014 as training set

The results can be found in the following table, along with the ones of our first attempt:

2012 to 2014	2013 to 2014	2014
58%	57%	63%

Table 1: Accuracy values depending on the training dataset

Although we expected that increasing the training dataset would guarantee a better precision, it seems that including previous seasons adds more noise than anything else. This can be explained by the fact that rosters changing each year implies that the most recent season will be the one with the most players still taking part in the tournament in 2015.

5 Conclusion

Thus, building a hypernode graph seems to be an efficient way to predict game outcomes and model individual skills in a 2-granularity problems. Although the accuracy of our model was not so good, several areas of improvement can be identified.

First of all, the lack of real rosters list probably biased the results, since the rosters collected through scraping Fox News website contained also the substitutes for each team, and the algorithm then written only kept the first 5 players of each teams. However, it is safe to assume that a highly skilled player is more likely to stay in the team several years, and he might not be considered by our algorithm.

A second obvious area of improvement comes from the fact that Basketball players have very distinct positions (point guard, shooting guard, forward...) that probably do not have the same marginal impact over the game. Thus, our assumption setting all individual contributions to 1 is probably wrong and may cause a bias.

Finally, the large amount of information provided for the Kaggle challenge leaves plenty of room for improvement by thinking of ideas to include this data. For instance, the fact that the game is played at home or not is probably impacting game results.