NBA Game Prediction by Using Graph Classification

MengYuan Shi, Austin Le

Department of Data Science
University of California, San Diego
San Diego, CA

January 2021

Abstract

When working with an NBA dataset, we wanted to figure out the best way to represent a network like structure amongst the teams and figured that the amount of time that each player spends on the court with one another would prove useful. By extracting this network and projecting player statistics upon each node, we will utilize GraphSage, a framework that will embed node features onto each player and aggregate each team to predict whether or not they can make the playoffs.

1 Introduction

GraphSAGE proposes a framework that can identify the node's neighborhood's structural properties as well as its role in the overall graph.[1] We can use GraphSAGE to utilizes node features to learn a general embedding function that simultaneously encodes the node's neighborhood's structure and feature distribution.[1] Additionally, GraphSAGE will be compared to graph convolutional neural networks on image data in the field of computer vision which just special cases of graphs. GCN apply simple "convolutions" to graph data to learn predictive features from weighted sums of progressively more complicated nonlinear relationships between nodes. GCNs share weights among nodes in a graph to more easily learn their inherently uneven distribution. GraphSAGE allows GCN to be applied in an inductive, unsupervised manner with trainable aggregation functions, which facilitates machine learning tasks on graphs that are not fixed and necessarily evolve over time, such as social networks.

In this paper, we are investigating the social network for individual NBA players and the relationship between each team. We will try to solve how to implement GraphSAGE to learn over the NBA players' dataset and find a correlation between teams that make the playoffs and those that do not. We will then run this model across current NBA players and their season's statistics to see which team is predicted to win given their current lineup. Our dataset is suitable for our given application because its graph has sparse and high-dimensional but meaningful connections that we can compress down to a low-dimensional embedding. For our current problem statement, we will be performing graph classification. The graph will have a total 120 data-point and it contains a set of node features. For each season we will have 30 data points (represented by 30 teams) and we decided to use 4 seasons in total to do our training model. Each node will be filled with players statistics as individual data points. We will use the team name as our node. Each node will contain the players of that team, with edges between players representing the amount of time they are on the court together. We will also label the data-point as 1 or 0 to indicate if they had made the playoffs.

The final goal is to use team statistics or players' statistics and analysis for predicting who wins games (by leveraging the players statistics and team statistics from the 2015-2018 season to predict the probabilty of making the nba playoffs for each team given their roster). In order to have enough training data, we choose the data from the 2015-2019 season (may use more data as we do the project) and since it's hard to predict the future season (do not have enough data for the current season) so we choose to predict the 2018-2019 season. We will also use data that we found to create edges between players on a team so that they will be connected by how much playing time they play with one another.

2 Datasets and Data Preparation

a) NBA_players.csv
This csv file contains all NBA players' statistics during 2016-2021 NBA season. And it contains feature such as player names, team, ppg, apg, rpg and so on. And we will use these features later as players' statistic for the training part.

b) NBA_teams.csv
This csv file contains all NBA Teams' statistics during 2016-2020 NBA season. And it contains the features such as team name, abbrev, year, playoff and standing(labeled 0 and 1 stands for didn't make to playoff and made to playoff). We will use these features later for the testing part.

c) events_[]-[]_pbp.cs
This csv file contains a historically account of the NBA play by play data by season. This dataset contains home players and away player IDS as well as the time during the game which well use to cross reference with player statistics. We have devised an algorithm that will utilize this dataset to determine the total

minutes that each player spends with one another. Our script will manually scrape the dataset from the website and read it in as a CSV, so the user doesn't have to deal with the large dataset.

3 Data Analysis

We spent lots of time on data scraping since the datasets we use are very complicated. It's important for us to decided what features vectors should we use as players' and team's statistics. And we will use those features later on.

When thinking of what makes an NBA Team great, we looked through different statistics and we settled with individual points-per-game, assists, rebounds, field goal percentage minutes and games played. The aggregated totals of each player on the team in conjunction with the amount of time played together would help us in determining how successful a team's network of players can be. This was our initial datasets where we collected individual player statistics and we had another with just the teams, a boolean value on whether or not they made the playoffs.

A main chunk of our project then revolved around cleaning and extracting the information that we wanted from a large play-by-play dataset that consisted of each and every NBA game in a single season. Depending on the team network that had to be created, what we tried to do was to gather all the games that they participated in and analyzed at what time did players sub out for one another. By using python/pandas we were able to create an algorithm that would create an edge between every player in the lineup and adjust its weight according to the time in the game. Once we collected all edges from every game in the NBA season, we grouped the duplicate edges and summed them to get a total amount of playing time between each player on the team. This would represent the edges between players that we needed to create each individual team's network.

4 Methods

4.1 GraphSAGE

GraphSAGE that allow Mechine Learning methods to effectively utilize the graph data. With the GraphSAGE, it no longer needs to take extra time to train a new embedding for a graph seen only at run time that might need to be classified quickly. The GraphSAGE embedding learned from a large, representative training graph would be serviceable for the given task.

4.1.1 Embedding generation algorithm

We first will use GraphSAGE's embedding algorithm to embedding the nodes in the case where the whole graph, we import a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and vector representations of node features $\mathbf{x}_v, \forall v \in \mathcal{V}$ and outputs embeddings $\mathbf{z}_v$ for every node in the graph.[1] For our graph classification graph, we are using 30 nodes which represent the 30 NBA teams as individual data point and each node will filled with players' statistics. There will be no edges between the nodes. For each node, it will contain 15 players of that team with players' statistics as their features, and we will use the amount of time players are on the court together as edges to connect each player.

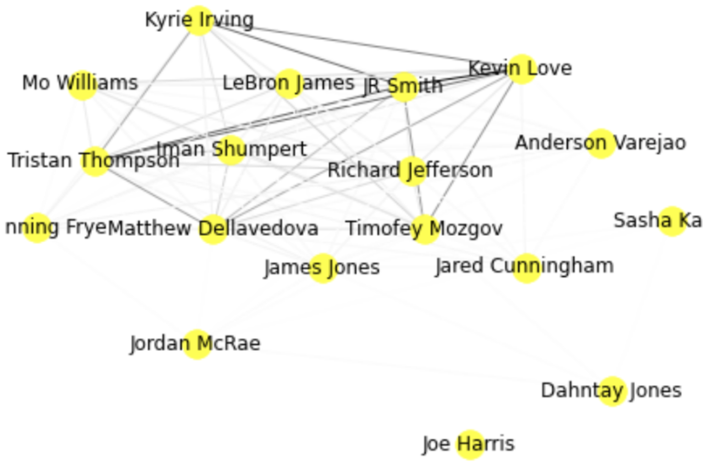
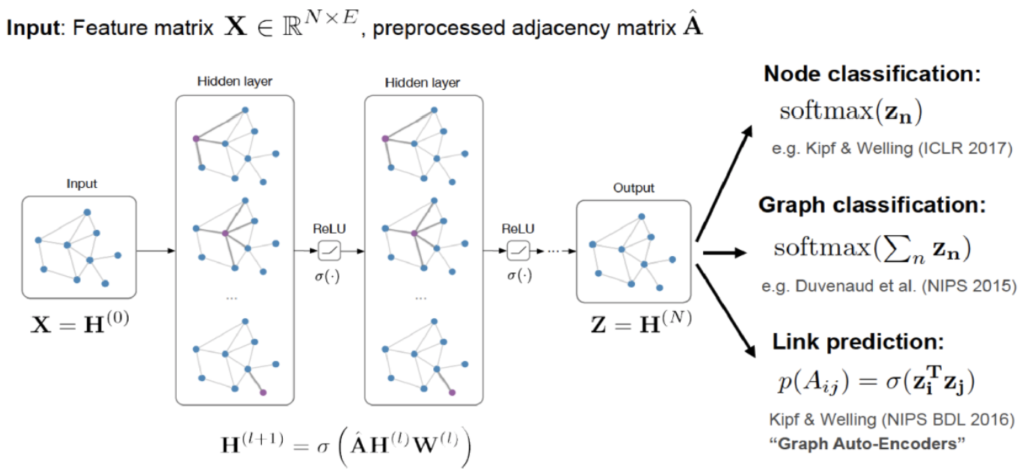


Figure 1: Graph Classification example for Cleveland Cavaliers' 2016 season

Based on the example in Figure 1, we can see each players in the team connected with each other and amongst all the links, the darkest 5 paths show that these players possess a larger weight in the amount of playing time with one another. As a result, it can indicate who the main 5 players in that team are. Going off of this example, we can see the likes of Kyrie Irving, Tristan Thompson, Lebron James, Kevin Love and J.R. Smith being a large presence within this team. We believe that this unique combination of mutual playing time and categorical statistics will allow GraphSage to utilize its aggregation to the fullest extent and control its weights based off the impacts that each player has within their team's node.

4.2 Loss Function

In order to get a clear result of our model, we use loss function to work on graph data and uses stochastic gradient descent to propagate error back through the network, progressively learning better and better representations even in the absence of labels. The output of the Softmax function will give us a vector, it standard for the probability distributions. The output will appended to the last layer of our graph classification network which will be the last layer of our GrapgSAGE model. The softmax calculation in addition to taking the negative log-likelihood over the data, and an early mistake in our implementations was including a softmax layer when it was in fact unnecessary [6].



Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$
Input
Hidden layer
Hidden layer
ReLU
$\sigma(\cdot)$
ReLU
$\sigma(\cdot)$
Output
$\mathbf{X} = \mathbf{H}^{(0)}$
$\mathbf{Z} = \mathbf{H}^{(N)}$
$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$
Node classification:
$\mathrm{softmax}(\mathbf{z_n})$
e.g. Kipf & Welling (ICLR 2017)
Graph classification:
$\mathrm{softmax}(\sum_n \mathbf{z_n})$
e.g. Duvenaud et al. (NIPS 2015)
Link prediction:
$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$
Kipf & Welling (NIPS BDL 2016)
"Graph Auto-Encoders"

Figure 2: We use this loss function to implement the Graph Classification and we use the method softmax to calculate the accuracy of our models

5 Results

We report results of graph classification prediction performed by our models on csv datasets. Our baseline models used a learning rate of 0.05 and we use epochs of 200 to calculate our model's accuracy.

Hyperparameter	Number
L2 reg	0.005
Learning rate	0.01

Table 1: Summary of hyperparameters used during GraphSAGE

GraphSAGE helps to infer node label probabilities for unlabeled nodes by specifying what a neighborhood in the graph should look like and building a special model based off those neighborhoods to capture both local and global structure of node features. GraphSAGE builds an impressively versatile model

	Loss	Accuracy
Training	0.0024	0.99
Validation	0.142	0.62

Table 2: Results of node prediction on Cora dataset

with the ability to be applied to multiple graph machine learning tasks, classify data using far less labels, and can be trained on large-scale dataset, as having training data into memory is not required for prediction. Its learned aggregation functions allow it to create node embeddings that range in specificity, learning to recognize simpler or more complex relationships depending on the task at hand. Graph classification use GraphSAGE can make predictions with a simpler model using lower dimensional feature representations extracted from a graph.

6 Conclusion

In this paper, we have outlined graph classification of nodes in graph data. GraphSAGE identifies important node communities whose features can help learn representations of never before seen node distributions. We thought that utilizing a network of NBA players would sort of create a more emphasized community of about 5-8 core players within a team and place less emphasis on the role players. It can be applied to completely unlabeled data in addition to being a practical semi-supervised approach that enables analysis of large scale, dynamic graph datasets like social networks. For our report, we use GraphSAGE to embedding the neighborhood and aggregate each team in order to predict which team can go to the playoff.

7 Future Work

Since the GraphSAGE, the node's neighbors do not have a natural ordering, we should do the vector operation over the unordered set. The aggregator algorithmic is able to maintain a high representational capacity since it is symmetrical. That allows the neural network model to be trainable and can do the node neighborhood features in an arbitrarily ordering.

Pooling aggregator:

For the pooling aggregator, it is also symmetric and trainable, the pooling approach is parametric and feeds each node's neighbor vectors through a trainable fully connected layer and then pools them using a max operation. This is then concatenated with the original node's vector and sent through the neural network layer and nonlinearity to become the node's representation in the next iteration. Unlike mean aggregator, pooling requires the network to take time to learn extra weights and biases with the benefit of creating more custom embeddings in the case of a more specific task. Through these aggregation functions, we can train our GraphSAGE model to learn the optimal way to compute a weighted average or pooling of node features, while at the same time learning to generate node embeddings that well represent graph data.

In terms of the coding portion, we were not able to fully represent and get the mean aggregates of a teams nodes. So currently running our Graphsage model would try to predict each player on the team and whether or not they made the playoffs, but for future work we would like to incorportate the mean pooling aggregator so we can change the dimensionality of all players into a single team node.

Pertaining specifically to our project, we believe that there were other factors that we could've changed in order to produce a higher accuracy. The biggest issue itself was the data that we used from eighthirtyfour; because it didn't come from an official NBA source, there were a couple things that we had to manually revise. The algorithm that we created had to be revised multiple times because of errors that were seen within the data. In addition, although we did our best to clean the data that we needed to cross reference, there may have been names that were misspelled or have a specific punctuation that just didn't match. We believe that utilizing a playing time network and statistics proves to be a nice proof of concept to build off of, and with some cleaner data there are definitely ways that would improve our utilization of GraphSage.

8 Responsibility

MengYuan Shi:Responsible for the paper researching and writing the report part as well as the visualization.

Austin Le: Responsible for the data cleaning and data scraping as well as the coding part.

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. (2018). Inductive Representation Learning on Large Graphs.

[2] Hongwei Wang, & Jure Leskovec. (2020). Unifying Graph Convolutional Neural Networks and Label Propagation.

[3] Relational Dataset Website. Retrieved from https://eightthirtyfour.com/data.

[4] Relational Dataset Website. Retrieved from https://www.basketball-reference.com/leagues/NBA_2021.html.
https://www.basketball-reference.com/leagues/NBA_2020.html.
https://www.basketball-reference.com/leagues/NBA_2019.html.
https://www.basketball-reference.com/leagues/NBA_2018.html.
https://www.basketball-reference.com/leagues/NBA_2017.html.
https://www.basketball-reference.com/leagues/NBA_2016.html.

[5] T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2016.

[6] CrossEntropyLoss. Pytorch.
https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html.

Appendix

A. Project Proposal

MengYuan Shi, Austin Le

B01_Capstone_Group_06

For our project, We will investigate the social network for individual NBA players and the relationship between each team. The goal of this project is to: use team statistics or players' statistics and analysis for predicting who wins games (by leveraging the players statistics and team statistics from the 2015-2019 season to predict the winners of NBA games for the 2019-2020 season). In order to have enough training data, we choose the data from the 2015-2019 season (may use more data as we do the project) and since it's hard to predict the future season (do not have enough data for the current season) so we choose to predict the 2019-2020 season.

For our domain this quarter, we discussed graph analysis and learned methods such as GCN, node2vec, random walks and so on. For the first week we learned how to measure different kinds of centrality of the graph. For that, we can use the degree centrality to get the top player prediction as part of our project. A player's node will generally gravitate as the "center" of it's team because they are considered a team's core that they will build around. This topic is interesting because we are not investigating the prediction of nodes and their labels, but rather creating a social network of our NBA players dataset. We will use this created network to compare against other teams and see what averages of our selected statistics indicate a more successful team. There have been other forms of prediction regarding NBA players and their standings, as well as betting, but none have utilized Neural networks to produce their results. In the replication project, we are focused on passing information through a network and trying to predict labels based on these aggregated features. For our project we want to do sort of a reverse, where we construct a graph based on features from past seasons and use this network to learn over what aggregated features can help produce an NBA championship caliber team.

For the methods that we attempt to solve the problem will be:

1) GraphSAGE: we use players' statistics and team's statistics from the 2015-2019 season and use those statistics to create a graph classification. The graph will have a total 120 datapoint and it contains a set of node features. For each season we will have 30 data points and we decided to use 4 seasons in total to do our training model. For player statistics we will use PPG, AST, TRB, MP, GP and so on. 30 teams will each be filled with players statistics as individual data points. We will use the team name as our node and there are no edges between

each team. And for each node, it will contain the players of that team, edges between players will be the amount of time they are on the court together. We will also label the datapoint as 1,2,3,4,5: finals, conference, semi, playoffs, none. (shows how far the team can go in the nba season)

2) GCN: Using previous nba seasons and the corresponding champion, we will use a GCN to learn over the nba players dataset and create a correlation between the nba champions. We will then run this model across the current NBA players and their projected stats to see which team is predicted to win.

3) We believe an implementation that revolves around Label propagation will be useful when finding comparisons between different teams. Just as an overall goal, teams try to find special "unicorn" and "generational" talent for their team and add other players that would help compliment their playstyle. In terms of a graphical representation, we believe that we could use label propagation amongst teams to determine their neighborhood characteristics and use this to compare to previous successful teams. If they have similarities in makeup, we would see that they may have a shot at the title for the upcoming season.

We would like to produce a report style paper where we will analyze the nba datasets that we will scrape ourselves. We also want to provide visualizations of interesting findings about NBA teams and their characteristics that have separated them from the rest of the teams as winners. If we are not able to find some sort of standings at the end of the season, we will try to predict just one overall champion. Additionally for a stretch goal, we believe that creating some sort of website that will allow user interactions to pick certain players and teams and display their most important characteristics would be an interesting challenge!