NBA Prediction: Traditional Data vs Advanced Data

Raul Alcantar
Computer Science
Stanford University
raulalc@stanford.edu

Peter Shih
Mechanical Engineering
Stanford University
ps306113@stanford.edu

Yu-Hao Han
Materials Science
Stanford University
hyh8871@stanford.edu

Abstract

The scope of this project is to create a deep neural network to predict the outcome of NBA games using both players' traditional data and advanced data. In the previous works about NBA games prediction, team statistics are the most used data. What makes this project novel is the use of data from every single player as the input layer, instead of team statistics or averaged players' stats. Additionally, there is no previous deep learning project working on NBA advanced data. We trained our model with advanced data and compared the performance and training strategy to that of one that would train on traditional player data as well as comparing it to models that trained on traditional team statistics.

1 Introduction

The National Basketball Association is the largest and the most valuable basketball association in the world, generating combined revenues of around 6.5 billion U.S. dollars every year. Aside from the huge revenue, NBA also generates a gigantic amount of data, including players' data and team's data. The data is highly beneficial to the teams because they can determine their strategies and game plan through statistics and analytics. In recent years, traditional data like points per game or rebounds per game are not enough to evaluate a single player's performance. The NBA's analysts create a list of advanced data which are calculated to interpret a player's true contribution to the outcome of games. For example, player efficiency rating (PER) is one of the most commonly used advanced metrics to measure a player's per-minute productivity. It adds up all the positive contributions a player makes to his team while subtracting the negative ones. It also adjusts for both pace and playing time to make it easier to compare players to one another. Advanced data are highly used in NBA analytics now. However, are they really more correlated to the outcome of games? We decided to investigate this more through machine learning.

2 Related Work

There have been a few models created by others that predict the outcome of the NBA, but the main information they look at is a team's season performance such as win percentage and their traditional team averaged statistics such as points or assists in a game or elo rating. Some of the inspiration we got for this project was from work done by Rohan Agarwal, Swapnil Ahlawat, Susmit Wani, and Wandan Tibrewal from Birla Institute of Technology and Science in Goa, India [1] where they used some of the intuitive team statistics along with player statistics, which is what caught our attention. They incorporated player statistics by getting a weighted average of features related to players of a particular team and this is what we wanted to focus on. Instead of getting a weighted average of the features and using that in combination with team statistics, we wanted to see what the impact of these player statistics alone independently would have on the outcome of the match.

3 Dataset Details

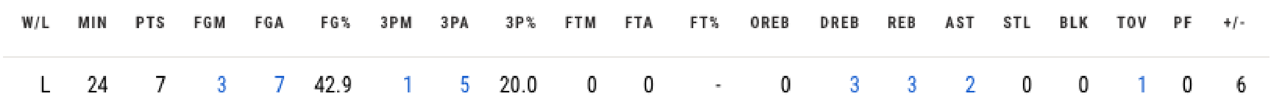
Since our project focuses on the individual player's performance and how it impacts the outcome of any given game, we had to look for data that clearly showed their personal stats for games spanning the previous 15 years. We eventually settled on the NBA's official website [2][3] after careful analysis of numerous websites since they are the official website of the league that will have the most accurate and up-to-date information and statistics for any given year, matchup, and player.

Once we decided where our source of data would be, we then worked on scraping the data manually off of their player box score tables. We wanted to compare the effectiveness of “traditional” player statistics to that of “advanced” player statistics and the NBA official website provided all of this information. The “traditional” player statistics consist of the normal information about the game such as whether the team won or lost the matchup, and whether the player was on the visiting or the home team. However, the data we focused on for our model were statistics such as the number of points, assists, rebounds, field goals, made, and field goal attempts, for a given game, along with many other statistics (full list of statistics is listed in Figure 1). The “advanced” player statistics consist of some similar statistics as the “traditional” ones such as whether the team won or lost, the matchup, and whether the player was on the visiting or home team, but the rest of the statistics differ. The other information they include is the player's offensive rating, defensive rating, net rating, and assist percentage, for a given game (full list of the rest of the statistics is listed in Figure 2). There are many more statistics that are included that will be explained later once again.

The NBA official website only provides the statistics for the given game, so we averaged out the statistics between all of the previous games leading up to a given game to get the player's overall performance in a certain season prior to the given game. We did this by accumulating the statistics and averaging the total number of matches played prior to a game. We used this approach because this would be the most realistic set of information available for future games. If one wanted to predict the results of a match that is occurring tomorrow, one would not be able to use the players' statistics for this season if it has not ended, however, they would have the statistics for all of the games leading up to tomorrow's game.

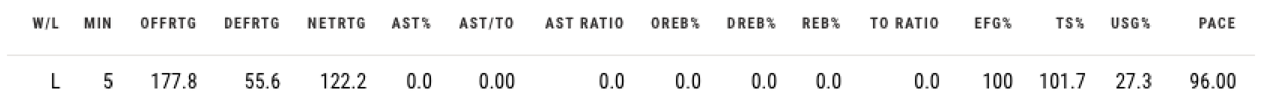
In addition to this, we also modified the data to clearly represent whether the team was on the home team or the visiting team and separated the statistics for the two because this is another feature that may differ in relevancy and impact on the outcome.

Once all the data was collected and put into a single file, we then reorganized the data to create a proper dataset for match-ups so that we would be able to feed it into the model. For every game, the number of players that played in that match can vary from 16 total players to 20 and because this was not consistent, we decided to only consider the 16 players with the most amount of minutes averaged thus far in order to keep the input for the model the same. Therefore for a single game, we recorded the outcome of the game and used that as our predicted value, y, and we concatenated every player's statistics into one example, x, so for an example using the traditional statistics, an example had $13 \cdot 16 = 208$ input features and an example using the advanced statistics had $14 \cdot 16 = 224$ input features. Since there



W/L	MIN	PTS	FGM	FGA	FG%	3PM	3PA	3P%	FTM	FTA	FT%	OREB	DREB	REB	AST	STL	BLK	TOV	PF	+/-
L	24	7	3	7	42.9	1	5	20.0	0	0	-	0	3	3	2	0	0	1	0	6

Figure 1: NBA Traditional Statistics Example



W/L	MIN	OFFRTG	DEFRTG	NETRTG	AST%	AST/TO	AST RATIO	OREB%	DREB%	REB%	TO RATIO	EFG%	TS%	USG%	PACE
L	5	177.8	55.6	122.2	0.0	0.00	0.0	0.0	0.0	0.0	0.0	100	101.7	27.3	96.00

Figure 2: NBA Advanced Statistics Example

are so many input features, we believed that a multilayer neural network would be the best choice as it would allow for the model to learn intricate features and predict the outcome with higher accuracy.

4 Method

For our model, we use PyTorch to construct two multilayer neural networks, with one using the "traditional" player statistics, and the other one with the "advanced" player statistics as input features. We used ReLU as the activation function for the hidden layers and sigmoid as the activation function for the output layer. The optimizer we utilized is the Adam optimizer with default parameters. As for the loss function, we used many to see which would lead to the highest performance. We use the binary cross entropy loss function, as well as the mean squared error loss function and the negative log-likelihood loss function. We then tuned the hyperparameters, such as the number of layers, layer size, learning rate, mini-batch size, regularization parameters, number of epochs, etc. of our model with a nested for loop.

5 Traditional Data Model

5.1 Baseline

For the traditional dataset baseline we built a basic 1-layer logistic regression binary classification model with sigmoid as the activation function and 16 total neurons using PyTorch. The baseline model achieved a training set accuracy of 93.4% and a dev set accuracy of 63.7%.

5.2 Experiments

We initially set our model for training the dataset to have 3 hidden layers with the size of [16, 24, 4] for the layers, and used it to first tune the batch size and the learning rate. We discovered that a batch size of 256 and a learning rate of 0.01 achieved the best balance between run speed and accuracy. Moreover, the train and test accuracy, as well as the loss, all converge around 500 epochs. To find the optimal number of layers, we ran several experiments on different network sizes using 3, 4, and 5 hidden layers. We discovered that 3 hidden layers are fully capable of training the model, and adding more layers had little to no effect on the dev set accuracy, but requires significantly more run time. As a result, the below experiments are done with 3 hidden layers with a batch size of 256, a learning rate of 0.01, and 600 epochs.

To find the optimal network size, we ran nested loops and looked at the performance of different network sizes. The program loops over a first hidden layer size between 16 to 48 with a step of 8, a second hidden layer size between 10 to 30 with a step of 4, and a third hidden layer size between 2 to 18 with a step of 4, which in total make up 150 combinations of network sizes.

With a 3-layer network, the model is able to achieve an average dev set accuracy of 70.2%. However, the train accuracy of the model is 78.7%, giving an average variance of 8.5% (dev accuracy - test accuracy), which indicates the model is overfitting the data. To solve this problem we added a dropout rate of 0.2 and ran the experiment again on different network sizes.

When applying dropout with a three-layer network, the model is able to achieve an average dev accuracy of 69.4%, train accuracy of 70.2%, with an average variance of just 0.27%.

5.3 Results

Among the experiments we found the optimal network size to be [16, 26, 6] with 0.2 dropout applied. Both the training and dev set accuracy are 70.1%. See Appendix Fig.1 plot for Loss vs Epochs for the Traditional Data Model.

Common methods of predicting NBA games using only team statistics usually have an accuracy between 66% and 72%. Our results show that the accuracy by using the stats of individual players on both teams to train the model is on the higher end of the spectrum. The model is able to perform well because it is able to learn how a single player can affect the outcome of the game. On the other hand, since we do not look at any team statistics or averages, the model isn't able to learn the strengths and weaknesses of specific lineups. Some players perform better when playing with specific teammates and poorly with others and using an individual's statistics is not able to capture that, thus leading our model to not perform higher than the 72% benchmark some models have set.

6 Advanced Data Model

6.1 Baseline model

As there was no previous game prediction article working on NBA advanced stats, we started the experiment based on what we learned from the traditional stats model, which is a simple 1 layer neural network with 16 neurons, 0.01 learning rate, 256 mini-batch size, 600 epochs, and no regularization. The result showed a huge variance with 90.7% training set accuracy and 64.7% dev set accuracy.

6.2 Experiments

We believe the overfitting issue mainly stems from the lack of number of layers because the input size for advanced data is large with 224 input features. Therefore, we ran the same three hidden layer experiments mentioned in Section 5 (Traditional Data Model). Within the 150 sets of hidden layer sizes, [48, 10, 14] shows the best dev set accuracy of 69.7% along with 73.7% training set accuracy. The difference between training set accuracy and dev set accuracy drastically dropped from 26% to only 4%.

To evaluate the model with a deeper network, we built a 4-layer network on the best 3-layer network, [48, 10, 14], by looping over the first hidden layer size between 48 to 96 with a step of 8. The best 4-layer network is [80, 48, 10, 14], but its dev set accuracy is still 69.7%. Therefore, we decided to work on a 3-layer network with a size of [48, 10, 14] to save computational time and resources.

As for learning rate and mini-batch size, we conducted experiments on 7 different learning rates [0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001] and 3 different batch sizes [128, 256, 512]. The model with

0.01 learning rate and 256 batch size still performs the best.

6.3 Results

From all the experiments we conducted on the advanced dataset, the optimal model is a 3-layer neural network with [48, 10, 14] hidden layers size, 0.01 learning rate, 256 mini-batch size. The highest dev set accuracy is 69.7%, which is close to but still lower than our results from traditional stats. See Appendix Fig.2 plot for Loss vs Epochs for the Advanced Data Model.

Generally, NBA advanced data is generated from a linear combination of traditional data. For example, offensive production rating is calculated from a linear combination of points produced, individual possessions, points per game, free throw percentage, 3-point percentage and field goal percentage. Therefore, conducting deep neural networks on traditional stats and advanced stats show similar capabilities of predicting the NBA game outcomes. The true values of advanced data are to help NBA analyzers evaluate the performance of players more accurately and objectively. But when it comes to predicting game outcomes, advanced data can not contribute more information.

7 Conclusion & Future Work

In this project, we developed a deep neural network to predict the NBA game outcomes from individual players' data. Our three layer network performed a dev set accuracy of 70.1%. The results prove that even though the model trained by only individual players' data is on the higher end of the typical NBA prediction accuracy range, the lack of team's data constrains the model's performance. As for advanced data, our three layer model achieved a dev set accuracy of 69.7% because advanced data are also derived from the traditional data so that it does not provide more information to the model, hence contributing little to the performance.

The future work is to combine the traditional players' and team's statistics and see how they could improve the model. In addition, different types of models can be made with algorithms and activation functions different from the ones we used, and also continuing to play with the hyperparameters could lead to more and more success potentially. These predictor models can then be implemented and tuned for more specific scenarios such as playoff matches as the circumstances and environment in those types of matches vary greatly from a regular season match. These predictor models could also be potentially used in the NCAA for college basketball matches or during March Madness with the use of transfer learning.

GitHub Link
https://github.com/raul-jigs/NBA-outcome-predictor

References

[1] Agarwal, Rohan, et al. “Prediction of NBA Games Using Machine Learning.” GitHub.com, 13 Feb. 2021, https://github.com/swapnil-ahlawat/NBA_Game_Predictor/blob/main/NBAPrediction.pdf.
[2] National Basketball Association. “NBA Official Players Traditional Boxscores.” *Players Box Scores | Stats | NBA.com*, National Basketball Association, 2022, https://www.nba.com/stats/players/boxscores.
[3] National Basketball Association. “Players Advanced Box Scores Traditional: Stats.” *NBA Official Players Advanced Boxscores*, National Basketball Association, 2022, https://www.nba.com/stats/players/boxscores-traditional.
[4] Weiner, Josh. “Predicting the Outcome of NBA Games with Machine Learning.” Medium, Towards Data Science, 15 July 2022, https://towardsdatascience.com/predicting-the-outcome-of-nba-games-with-machine-learning-a810bb768f20.

Appendix

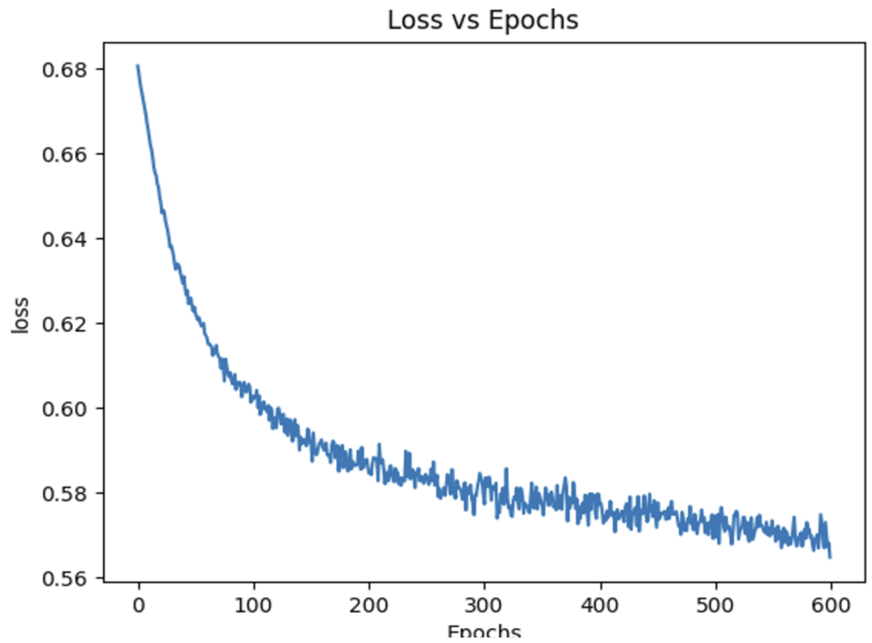


Fig. 1 Traditional Data Model Loss vs Epochs

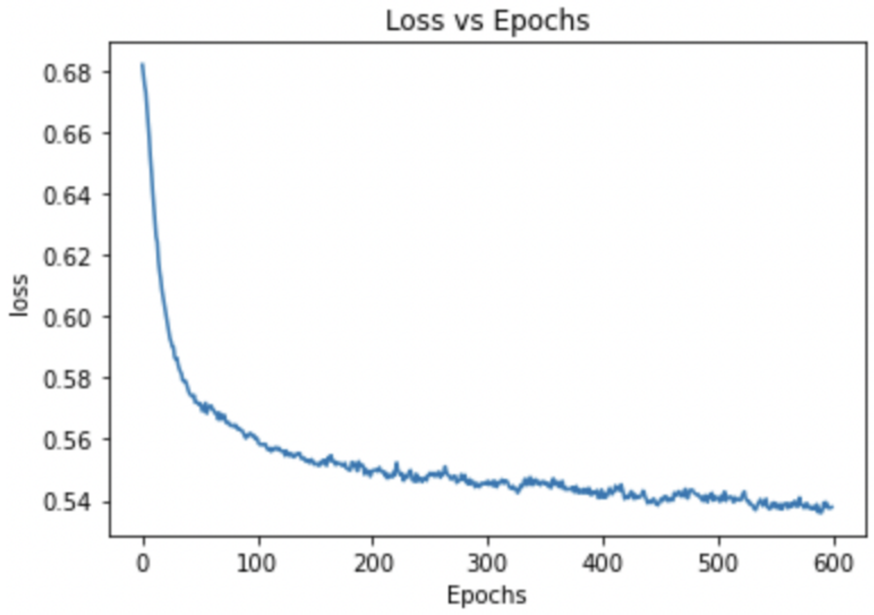


Fig. 2 Advanced Data Model Loss vs Epochs