Using a Deep Neural Network to Predict NBA Wins and Losses

Rebecca Andrews, Thomas Possidente

MOTIVATIONS

As lifelong athletes and current varsity athletes at Vassar College, we have been exposed to the world of sports statistics early and often. There are times where the performance of certain teams in a season seems to nearly guarantee their victory in a particular game, but this is more based on anecdotal evidence than anything else. The question is then, how much of sports is algorithmically predictable and how much cannot be captured in hard numbers? What statistics are the best predictors of performance? Moreover, can we beat the best professional sports gamblers using a deep neural network to predict wins and losses? We set out to answer this last question directly, while peripherally addressing the first two.

METHOD

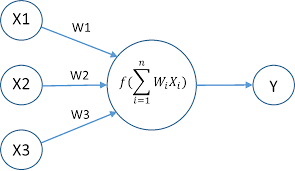
DATA GATHERING

All of the NBA basketball data used for our neural network was found on <https://www.basketball-reference.com> from the 2017-18 or 2016-17 seasons. We were able to scrape (gather) this data using rvest, a scraping tool written for R. Individual game data was organized and aggregated to calculate the cumulative statistics at every point in the season for each team. This was done so that for each matchup we could use the season stats (so far) for each team as the input. We felt this made sense because this would be the information that is available if one was using the neural network to predict who would win a future game. We calculated cumulative stats for field goals per game, field goals attempted per game, field goal percentage, three pointers per game, three pointers attempted per game, three point percentage, free throws per game, free throws attempted per game, free throw percentage, offensive rebounds per game, defensive rebounds per game, total rebounds per game, assists per game, steals per game, blocks per game, turnovers per game, personal fouls per game, and points per game (a total of 36 stats). 987 data points were gathered for the training portion and 488 data points were used for the test portion.

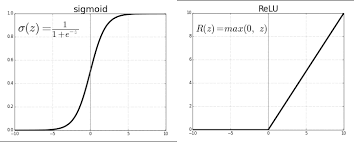
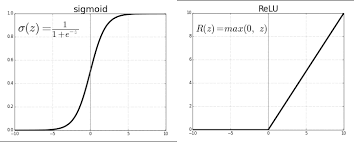
In addition, we also gathered basketball data for basketball teams in the Liberty League from <http://libertyleagueathletics.com/index.aspx?path=mbball>. This was done to see if our neural network could generalize outside of NBA data, and only a small set of data was gathered here. 7 data points were used.

DESIGN OF THE NEURAL NETWORK

Our supervised deep neural network (DNN) was created on a Google Colab notebook using Keras. A DNN is a multilayered neural network in which each layer is a group of nodes that receive inputs from nodes in a previous layer, perform a computation on the inputs (figure 1), and output an activation value to nodes in the next layer. Learning occurs in such networks by training the network with many examples and comparing the network’s output to the correct output, then changing the values of the weighted connections between nodes to reduce the error. From this iterative process of weight refining called “back propagation”, the network eventually settles on a set of connection weights that result in something close to the correct output given an input of the same type it was trained on.



**Figure 1** shows how a single node sums the products of all inputs (X) and their corresponding weights (W) and uses an activation function (f) to compute a single output.

Our network contained one input layer, four hidden (intermediate) layers, and one output layer. 36 statistics were fed into the input layer for each of two teams (for a total of 72 input nodes). The output was the predicted probability that the away team would win. Each hidden layer used the ReLU (Rectified Linear Unit) activation function (figure 2a). The ReLU is widely used in DNNs and were used here because they tend to accelerate training compared to other activation functions like the sigmoid and tanh functions (Krizhevsky et. al., 2012). The sigmoid activation function (figure 2b) was used in the output layer (consisting of a single node) in order to make the output a number between zero and one that represents the probability of the away team winning.

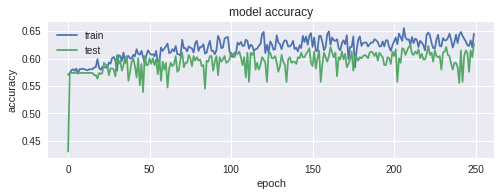
A B

**Figure 2** a) shows the rectified linear unit activation function while b) shows the sigmoid activation function

HOW TO RUN ON GOOGLE COLAB  
 Our Colab project file can be found at the following link: <https://colab.research.google.com/drive/1ssVNtbkHnieq4m9eWxd-QuZQygYJT9zR>. In order to run this file, several CSV files containing data need to be uploaded. All of the files can be found in the folder called “Key Files” in our github repository. The folder “ImportantCSVOrig” contains the training data, the folder “NewTestOrig” contains the test data, and the folder “LibertyLeague” contains the data from the Liberty League games. In our collaboratory file, each time one of these csv files should be updated there is a note specifying this. In total, there are 6 files that should uploaded to run the model.

RESULTS

If the model output was 0.5 or greater (50% or greater chance the away team wins) and the away team did win the game, this was considered correct, and if they did not win the game, this was considered incorrect. If the model output was less than 0.5 and the away team lost the game, this was considered correct, and if they won the game, this. After 250 epochs of 20 examples each, the model was able to predict wins and losses on test data (data that the model was not trained on) with ~60% accuracy (figure 3). Overall, this is relatively successful given the normal success rate for gamblers at closer to 50%.



**Figure 3** shows the accuracy of the model’s predictions of the data it is being trained on (blue) and the test data (green) that it is not trained on over 250 epochs of 20 samples each.

As an additional point of exploration, we compared the model’s predictions on the Liberty League data to the actual results for these games. The model predicted 7 wins for the home team, while in fact there were 3 wins, which is about 40% accurate. This seems to suggest that the criterias used to predict NBA games may not be completely generalizable to Liberty League data.

DISCUSSION

Professional sports gamblers need to win at least around 53% of the time to break even due to the bookmaker’s fee of around 3%. However, in order to make a living from professional gambling, it is estimated that you need to win around 55-58% of your bets (Bluth, 1997; Winning Percentages). Using our DNN, and only betting on NBA games, this could reasonably be achieved and possibly exceeded, as our model had a prediction accuracy of about 60%, This suggests that our model could be used to support a healthy gambling addiction. Using a similar methodology one could conceivably create a DNN for any sport with enough data to support robust training.

Our model does seem to suggest, however, that this particular model may work really well only for NBA data, and at the very least not very well for Division III basketball. Though we had a very small sample size, the fact that the home team was predicted to win in each case suggests that difference in magnitudes for things like points per game, rebounds, etc. between college and NBA basketball may have been too much for the model to generalize very well. Further study would increase the sample size of college data and continue to see if we continue to see a pattern of always selecting the home team.

In the current model, the test data is from the 2015/2016 season while the training data was from the 2017/2018 season. It is possible that the model would do even better if the training and test data was from the same season, but it is probably a better sign that our data was able to generalize across different seasons. This suggests that the patterns in the data that our model picks up on are patterns that are truly consistent across seasons. Had we kept the network up to date with the test data (training the network on every game of the season up to the game being predicted), we predict that the model would have exceeded 60% accuracy. Our model might also have been improved if we spent more time tuning the parameters or used some sort of optimization strategy to automate this for us.

Overall, our model successfully predicted game outcomes above chance and even above the normal basketball gambler. It was an excellent learning opportunity for us both, as we went from limited knowledge of neural networks to building our own!

References

Bluth, A. (1997, November 09). *Earning it: Life's a gamble. A few people make it a profession.* Retrieved May 6, 2018, from https://www.nytimes.com/1997/11/09/business/earning-it-life-s-a-gamble-a-few-people-make-it-a-profession.html

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097-1105).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

*Winning percentages*. Retrieved May 6, 2018, from http://www.professionalgambler.com/Winperct.html