Implementation/ Version Control Document

NBA game result predictor

Group DS4

by

Maciek Jankowski (4651294),
Ray Sambath (4665295),
Luke Chin A Foeng (4609972),
Paul Simon Schön (4531515),
Matteo Rebosolan (4664973)

Contents

1 Introduction

This document describes the various changes that happened during the project and how they were handled by the team.
As the team started with virtually no experience in the matter of machine learning, our first design plan for the game predictor turned out to be quite unfeasible, or at least beyond our level of knowledge. In particular, the structure of the neural network has been reimagined twice, as that was the component we struggled the most with.

2
First Iteration

The first design for the neural network had the player names as inputs: supposedly, the network would be trained to learn which individual players were the biggest contributing factor to their team winning or losing. However, we promptly ran into some issues:

- **Dataset**: The list of active NBA players is constantly changing and evolving, due to trades and players entering/leaving the league. This would mean that different datasets would have to be created for every different season and then updated with injuries/trades, making the whole process very inefficient and time-consuming.
- **Interpretation by Neural Network**: The players' names were to be input as strings, a data format not supported by Keras for basic machine learning and tricky to work with as we found out.
- **Not a good representation**: our inputs to the neural network are supposed to be predictors of the final result of a game; while knowing who is playing is important, it is not a good representation of how the teams actually performed in that game.

Due to the above reason the team decided that the neural network was to be redesigned from the ground up.

3 Second Iteration

The main problems from the first iteration of the neural network revolved around the inputs being strings. Thus, the team decided that using numbers instead would be the best thing to do. We want these numbers to be as representative as possible of a team's performance in the season, so we decided that the inputs for this new iteration of the neural network would split in two and then compared in a siamese neural network. Each of the two inputs consists of one team's stats up until the current game for the season, averaged per game.

The NBA keeps an extensive record of its games with hundred of statistics. We couldn't just include all of them in the model as that would simply increase the complexity of the neural network too much and would also require the generation of massive data sets. Therefore, we narrowed it down first to 20 and then to 4 stats (**FG%**: shooting accuracy, **TOV**: of turnovers, **ORB**: offensive rebounds, **FT%**: free throw accuracy).
These stats represent multiple aspects of the game: **FG%** and **ORB** measure how well a team does on offense while **TOV** and **FT%** take care of defense and drawing/staying clear of fouls, respectively. These stats are therefore a good compromise between completeness and simplicity for the model.

Again, problems started to arise during the implementation of the new neural network. First, an entirely new data set was to be generated, but this was expected. More importantly, we realized that the siamese network structure is ideal for highlighting the similarities between the two inputs, which is not what we were interested in. Thus, the neural network had to be redesigned again, although in not such a drastic way as the first time around.

4

Third & Final Iteration

This redesign is less drastic than the first one since we only had to reshape the input and not completely overhaul the formats of neural network and data set. To solve the problems related to the usage of a siamese network the team decided to simply merge the two inputs into one: this was done by using the difference in values of the 4 game statistics between the two teams. These stats were then normalized in a [0,1] range as that is a requirement for Keras neural networks. Changing the inputs did not require a big change in the data set as was the case for the first redesign so the team could implement and test the changes in the neural network much more quickly.

One thing that could still be improved is which stats to use as input. Altough we feel the ones we used are a good compromise, it may be that trying models with more statistics/different combinations will yield better results while keeping the complexity under control.