Design Plan

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

In our previous report, we briefly described the project development plan and the various steps we would take to get started on the project. In this report, we will be going more into the details regarding the software used, architecture and implementation of our project.

The aim of this project is to produce an algorithm capable of determining the winning team in a regular NBA season match, based on the starting lineup and data relating to the games played in the past. In order to achieve this a neural network will be designed and trained.

2 Design Architecture

Machine learning software is not a typical software in terms of architecture. There are only 2 sub-softwares which have to interact together. It's the data retrieving software and the neural network software. First is needed for the later, but the actual interaction between them is limited to requesting the data. As the learning of neural network takes part offline, there is no need for expensive server-client architecture. Whole software will work on a single machine. The architecture will be presented in form of layers:

- Back-end layer
  - Computing layer - Machine learning algorithm for a classification problem in form of feed forward neural network with multiple inputs - binary output
- Middle-end layer
  - Data acquisition layer - Web-scraping script to access and extract the data from online servers
  - Data processing layer - Retrieving only important information from the collected data, organize it in correct manner, save to .csv file, splitting into training and testing sets
- Front-end layer
  - Access to the model created in back-end layer, making a prediction using this model, obtaining the results of AI analysis

To ensure that the software is sustainable and testable, the middle-end and front-end layer have to be independent from the computing layer (each compilation in back-end layer is very resource consuming). As the software is not to be deployed for broad audience, there is no need for a fancy user interface. The goal of this architecture is to allow for parallel work on all the layers and decrease the hardware requirements of this software.

3
Web Scraping and Dataset generation

In order to generate the dataset for training the neural network large amounts of data need to be scraped from the internet. Fortunately years worth of NBA season data are available on multiple sites. The primary features used for the data set are field goal percentage, the number of turn overs, the offensive rebounds and the number of free throws. There is evidence that these features are the most prominent ones in determining the success of an individual team. If it turns out that this is not the case, it might be necessary to create a different data set in order to achieve higher accuracy.

The python libraries used for webscraping are mainly BeautifulSoupIV, Pandas and Urllib. They will be described in the following sections.

3.1. Urllib

Urllib is a model that provides several function and classes to open urls and request the html data of the website in question. It will mainly be used to provide an input for BeautifulSoup.

3.2. Pandas

The built-in pandas package in python will primarily be implemented to scrape certain tables off of the ESPN site. Pandas does this through the use of the readhtml function located within the pandas module. This outputs the tables located within the specified url. This outputs a list of lists, with one long string per list. These strings are then split up into smaller strings through the use of the built-in .split function. From here, another function is implemented to take out any unnecessary values and characters by taking the important values from the current list and placing them into another list that will then be replacing the original list. This will be done to each list within the set of lists the pandas function gave originally.
From here, these lists will now be appended into a comma-separated value (CSV) file. This is done through the use of the CSV module. From here, since the newline code ($\backslash n$) is still within the lists, new lists are made through the use of a for loop that searches for these symbols and creates new lists containing the items between them. These lists are then checked to see that they are, in fact, all of the same length. Once these checks are ran, the list of lists is then written into a new csv file named after the date during which the game was played as well as who was playing who. Additionally the use of the pandas dataframe makes it very convenient to manage, manipulate and organize large tables of data. This will be exploited as much as possible to generate the large dataset necessary for training.
The above process is then applied to every match during the 2019-2020 season as well as to the games that occurred during the previous 10 seasons.

3.3. BeautifulSoup

In order to scrape data from more complex websites where pandas cannot properly interpret the tables present, we rely on BeautifulSoup. BeautifulSoup relies on urllib to open and read the URL given to it. Once this is done, BeauitfulSoup is able to sift through this HTML for the correct data specified once the class in which the data is, is specified. A website is structured using HTML tags, the data is organized and displayed based on these tags and Beautifulsoup allows to retrieve the data by specifying the location based on inputting the HTML data and formatting. The HTML data is provided by Urllib. Once the data is found, it is then all placed into lists to later be placed into a CSV file once all of the irrelevant characters (such as the spaces, commas etc.) are removed. These CSV files will later be used to train to generate the data set, which is used to train the neural network.

3.4. Dataset generation

The dataset that is to be generated, will consist of two inputs x_1,x_2 and the output y for which the weights are to be determined. In our case the output is whether the home team wins, loses or draws in a game, this will take integer values between 1 and -1 respectively. The x_1 and x_2 will correspond to the home team and away team. The x and y data points will be scraped from the internet for each game of each season. The input features, are vectors consisting of the field goal percentage, the number of turn overs, the offensive rebounds and the number of free throws for each team. Experimentation is necessary to determine whether the past games should be factored into these values, by taking for example the average field goal percentage of the entire season or only of the last three games. Since these values are readily available after every game, for every game for every season of the last 10 years, availability of the data is of no concern. Therefore the data will be retrieved from the internet and manipulated using pandas dataframes, that make it convenient to compute several columns of averages for example over the course of a season. These can be stored in a csv file that will contain the x_1,x_2,y pairs necessary for training.

4 Neural Network design overview

4.1. Neural Network Layout

The software we decided to use to develop the neural network is Keras. Keras is a python library that uses TensorFlow as its backend. Keras and TensorFlow were chosen due to their suitability to classification problems (as is our case) and their extensive online documentation.

Since determining the outcome of a game is, again, a classification problem, the features of the neural network will have to be optimized for the task:

- **Type of neural network**: we will implement a neural network of the Siamese kind that will take as inputs two arrays containing average game stats for both teams. The network will work on the inputs separately through the use of one dense layer for each team and then concatenate the outputs of these layers to yield a comparison that will be processed by other layers to provide the final result.
- **Supervised learning**: we will be feeding the neural network the results of past games during the training phase; thus the learning process will be "supervised".
- **Backpropagation**: during the training phase, the model will make a prediction for some games and then compare this prediction with the actual result of the game in order to adjust the weights between the neurons and improve the accuracy of future predictions.
- **Loss function**: the cross entropy loss function works well with classification problems and we'll therefore prefer it in our neural network. The loss function for the output layer will be determined through experimentation in order to minimize the loss.

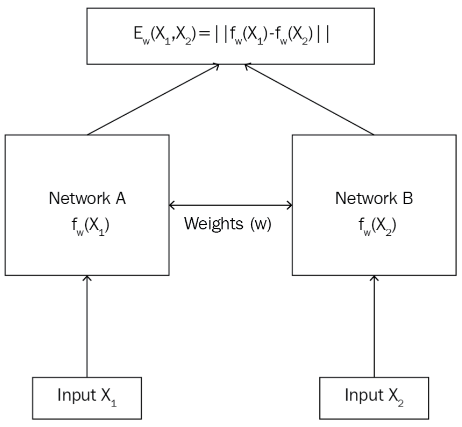

Figure 4.1: Structure of a Siamese network

The hyperparameters of the network, such as the exact number of layers and neurons, will be determine through experimentation during development as they are highly specific to the task and there is no clear cut way of determining them beforehand.

4.2. Hyperparameters

In order to achieve a functioning machine learning algorithm, the model parameters need to be determined. In terms of neural networks those are mainly related to the breadth and the depth of the network, how many hidden layers and how many nodes per layer. These parameters are referred to as hyperparameters. The setting of hyperparameters is closely related to the data and task of the neural network and is not obvious before hand. Heuristics and experimentation will be used in order to achieve a functioning model, however if necessary and if time permits it might be possible to make use of optimization schemes like genetic programming in order to achieve a more accurate model.

Genetic programming is an optimization method that attempts to replicate natural selection by selecting the fittest candidates out of a population, generating a new population based on the fittest candidates and introducing random variation in order to discover fitter variants. With respect to neural networks and hyperparameters, the aim is to create a population of neural networks with varying hyperparameters, selecting the most accurate network configurations, based on game prediction ability and then generating new neural networks based on the hyperparameters that define the most accurate neural networks. This approach is a lot more time efficient, than finding the optimum parameters based on comparing every single combination and will produce more accurate neural networks models than randomly or heuristically selecting parameters.

For the time being however we will consider one dense layer of 4 nodes per input that will be merged into a dense layer with 2 nodes. This may be subject to change.

5 Testing and Validation

There are several things to consider for testing and validation. First sufficient data needs to be available for training purposes, the more data the less likely it is to create a model that overfits. Secondly sufficient data needs to be available for validation in order to minimize the square residual error of the model, finally another set of data needs to be available to test and compute the accuracy of the network. Several approaches are considered to achieve this. Currently the aim is to train on six seasons worth of data, validate on two seasons and finally assess accuracy over the last two seasons. Another approach would be to train on 60of the games in one season, validate over 20and test on the remaining 20. The more effective approach to minimizing the square residual error, without overfitting will need to be determined experimentally. The prediction accuracy will show which approach produces a better model.

The reason 10 fold cross validation is currently not considered is because we assume that the order of time is relevant and that it would therefore not make a lot of sense to assess accuracy for earlier seasons. Depending on the effectiveness of this approach more season data might need to be scraped, or if it become apparent that time, does in fact not play a defining role, cross validation might be considered.

6 Summary

The intent of this project is to produce a piece of software, capable of predicting the game outcome of two NBA teams, by taking in the average past data relating to the field goal percentage, the number of turn overs, the offensive rebounds and the number of free throws for each team. This will be achieved by scraping all the necessary historical game statistics and generating a training dataset of these statistics and training a siamese neural network capable of determining the outcome with high accuracy. The libraries used for this are Urllib, BeatifulSoup and Pandas for webscraping, Pandas for Data wrangling and Keras and Tensorflow for the generation of a neural network.

Bibliography

Gerritsen, B., *WI3615TU – CS Minor Projects BigData and AI*, Delft: Delft University of Technology, Nov 2019.