Using Feature Weighting For College Basketball Game Prediction

Bryson Neel
Department of Computer Science
University of Texas at Arlington
Arlington, Texas
bryson.neel@mavs.uta.edu

Garrett Shimek
Department of Computer Science
University of Texas at Arlington
Arlington, Texas
garrett.shimek@mavs.uta.edu

Aditya Mishra
Department of Computer Science
University of Texas at Arlington
Arlington, Texas
aditya.mishra@mavs.uta.edu

***Abstract*—This document provides summary and discussion of three articles that will aid in our understanding and implementation of a machine learning program that will attempt to predict winning teams of basketball games. It also contains a detailed outline of our implementation of an algorithm used to predict winning college basketball teams.**

I. Introduction

For our Fundamentals of Machine Learning group project, we chose to create a machine learning program that would predict winning teams in college basketball games. To help us determine how to do this, we reviewed three articles related to machine learning sports prediction, and then we used our findings to create a program that would be able accurately predict winning teams. It utilizes the total stats from each team, as well as the data from each individual player.

II. Literature Review Of Articles

A. Article 1: Improving Sports Outcome Prediction Process Using Integrating Adaptive Weighted Features and Machine Learning Techniques

Authors: Chi-Jie Lu, Tian-Shyug Lee, Chien-Chih Wang, Wei-Jen Chen

In this article, the authors document their use of machine learning to predict NBA game results. They separate the process into six steps: data collection, data normalization, feature design based on adaptive weighting, prediction model construction, performance evaluation, and final results.

Data collection was the first step. The authors acquired data from https://www.basketball-reference.com, which contained every NBA 2018-2019 regular-season game. There were 1,230 regular-season games. Each game has a home team and an away team, so there were 2,460 data points in total. Each data point contained 15 variables, including 2-point/3-point field goal attempts/percentage, free throw attempts/percentage, offensive/defensive rebounds, assists, steals, blocks, turnovers, personal fouls, whether the team was the home or away team, and the team score. The first 14 variables were $X_{1,t}$ through $X_{14,t}$ and the last one was Y_t, where t meant that it was in the t^{th} game.

Data normalization was implemented so that larger input variable values would not influence small input variable values to reduce prediction errors. The equation used was:

$$X'_{i,t} = \frac{X_{i,t} - minX_i}{maxX_i - minX_i} \tag{1}$$

where $maxX_i$ and $minX_i$ are the maximum and minimum values of the attribute X_i.

Feature design based on adaptive weighting is next, and it "aims to integrate adaptive weighting techniques into feature construction" [4]. Feature construction is done using the interaction between game-lag information and the exponential power of adaptive weighting" [4]. Game-lag information is the nth game before game t. The adaptive weighting for this paper is "based on inverse-distance weighting (IDW) by integrating exponential power d as a weighting control parameter" [4]. The designed feature is:

$$\bar{X}^{l,d}_{i,t} = \Sigma^{l}_{n=1} AW^{l}_{n,d} \times X'_{i,t-n} \tag{2}$$

and adaptive weighting is:

$$AW^{l}_{n,d} = \frac{(l-n+1)^d}{\Sigma^{l}_{1} n^d} \tag{3}$$

where i = variables sequence ($1 \leq I \leq 14$), t = target game ($1 \leq t \leq 82$), $l = game - laginformation (3 \leq l \leq 6)$, d = weighting control parameter ($0 \leq d \leq 3$) [4].

$$\bar{X}^{l,d}_{i,t} \tag{4}$$

"is the designed i^{th} predictor variable at the t^{th} game with l game-lags based on d exponential power of adaptive weighting" [4].

So, if the first normalized vector (i = 1), l = 3, d = 1, for game 25, (t = 25), then the equation would be:

$$\bar{X}^{3,1}_{1,25} = \Sigma^{3}_{n=1} AW^{3}_{n,1} \times X'_{1,25-n} = AW^{3}_{1,1} \times X'_{1,24} + AW^{3}_{2,1} + X'_{1,23} + AW^{3}_{3,1} \times X'_{1,22} \tag{5}$$

where

$$AW^3_{1,1} = \frac{(3-1+1)^1}{1^1+2^1+3^1} = 0.5 \quad (6)$$

$$AW^3_{2,1} = \frac{(3-2+1)^1}{1^1+2^1+3^1}\ 0.34 \quad (7)$$

$$AW^3_{3,1} = \frac{(3-3+1)^1}{1^1+2^1+3^1} = 0.16 \quad (8)$$

and the weighting distribution when $d = 0$ would be:

$$AW^3_{1,0} = \frac{(3-1+1)^0}{1^0+2^0+3^0} = 0.33 \quad (9)$$

$$AW^3_{2,0} = \frac{(3-2+1)^0}{1^0+2^0+3^0} = 0.33 \quad (10)$$

$$AW^3_{3,0} = \frac{(3-3+1)^0}{1^0+2^0+3^0} = 0.33 \quad (11)$$

In the next phase, prediction model construction, 14 features and 5 machine learning methods are used to construct a prediction model. These methods are CART, RF, SGB, XGBoost, and ELM. Each variable "can be extended to 16 features which is the combination of four game-lags and four weighting control parameters" [4]. The 14 variables with game-lag and weighting control parameters "are used as predictor variables" [4]. This allows you to evaluate the effects that different game-lag and weighting control parameter values have on prediction. The final score of a game (Y_t) can be found using the equation:

$$Y_t = f(\bar{X}^{l,d}_{i,t}) \quad (12)$$

where $1 \leq i \leq 14, 3 \leq l \leq 6, 7 \leq t \leq 82, 0 \leq d \leq 3$ [4].

In the next step, performance evaluation, cross-validation is used "to estimate the performance of the proposed prediction process" [4]. The authors used root mean square error (RMSE) as a prediction performance indicator. It's calculated as:

$$RMSE = \sqrt{\frac{1}{n}\Sigma^n_{i=1}e^2_i} \quad (13)$$

"where n is the sample size, and e_i represents the error of predictions" [4].

For the final results phase, the best production process is chosen, and from that, the game-lag information and exponential power on adaptive weighting is chosen. The authors found that the best game-lag value was 4 and the best weighting control parameter value was 1 for all 5 ML methods.

This article provides us with a possible algorithm we can use when implementing our program, and at the very least, helps us understand how to interpret and manipulate the data as one of the chosen data sets is very similar to the data set used in the article.

B. Article 2: March Madness Prediction: A Matrix Completion Approach

Authors: Hao Ji, Erich O'Saben, Adam Boudion, Yaohang Li

This article presents its own method for predicting the outcome of the games in the NCAA Men's Division I Basketball Tournament, also referred to as March Madness. The authors' goal is to enter the March Machine Learning Mania competition. It is hosted every year for whoever can create the best prediction algorithm. Each submission is judged by the Log Loss function, so the authors' goal was to minimize the Log Loss for their submission. They list the function as:

$$LogLoss = -\frac{1}{n}\Sigma^n_{i=1}(y_i log(p_i) + (1-y_i)log(1-p_i)) \quad (14)$$

"...where n is the number of games, p_i is the winning probability of team 1 playing against team 2, and y_i equals 1 if team 1 wins over team 2 and 0 otherwise" [5].

The authors break the prediction down into a few different steps. First is matrix completion, then a neural network, and lastly, some probability adjustments.

The first step, matrix completion, utilizes the data from all the 2015 regular season games that were played. There were 364 different teams and only 3771 match-ups, so in order to create an exhaustive list of match-ups for any two teams matrix completion was used. There were actually 13 different matrices made, each comprising a different stat. They are:

1) field goals attempted (fga);
2) field goals made (fgm);
3) three pointers attempted (fga3);
4) three pointers made (fgm3);
5) free throws attempted (fta);
6) free throws made (ftm);
7) offensive rebounds (or);
8) defensive rebounds (dr);
9) assists (ast);
10) turnovers (to);
11) steals (stl);
12) blocks (blk);
13) personal fouls (pf);

These predicted matrices were only considered the starting point. They called the values in each matrix the predicted "performance accomplishments" for each team [5]. These "performance accomplishments" were then used in a neural network to make a guess at each teams' score for each match-up in the tournament, based on some historical records [5].

Finally, their last step was to make probability adjustments to their predicted scores. To change the scores into percentages they used a modified version of normalization. They raised

the standard normalization formula to the sixth power. They did this to emphasize any difference in scores between the two teams. Their example was a team winning 80 to 50.

$$p_{team1,team2} = \frac{80^6}{80^6 + 50^6} = 0.9347 \tag{15}$$

After this a few more equations were used to modify the predictions. One of them took into account the initial ranking(seed) that the team was given in the tournament. The equation is shown below:

$$p_i = \frac{1}{2}(\frac{1}{2} + \frac{3(seed_{team1} - seed_{team2})}{100} + p_i) \tag{16}$$

"...where $seed_{team1}$ and $seed_{team2}$ denote the rank of team 1 and team 2 in the tournament, respectively" [5]. For the last adjustment, they applied two simple corrections to the probabilities. Any game where a seed 1 plays a seed 16 will automatically have the seed 1 winning, giving it a probability 1 and the seed 16 team a probability of 0. Next, if the score separation for the two teams is greater than twenty points, the team in the lead will automatically have their probability set to 0.9545 because of the large lead.

The article ends by talking about some potential future improvements like including potential correlations between team seeds and their initial performance matrices. Also, they list "... more careful probability adjustments..." [5] as another improvement. These are some things we can take into consideration in our own predictions.

C. Article 3: The application of machine learning and deep learning in sport: predicting NBA players' performance and popularity.

Article Author: Nguyen Hoang Nguyen, Duy Thein An Aguyen, Bingkun Ma Jiang Hu
Article Link: https://www.tandfonline.com/doi/full/10.1080/24751839.2021.1977066?scroll=topneedAccess=true

This article is more about the way of describing or reporting the project in words. It gives us more information about how to write a report with visual presentation and conclude the project with accuracy. That's what we will implement in our project.

Data Preparation: In this article, they have used data-sets from 1979 whereas we will use data-sets from a single year. Since, there are not many regular data-sets available data about some players before. We will implement data from 2019 to predict future game results. In the article, whenever there is a missing value (no success or not present), they have used it as there was zero attempt from the player which is exactly what we are going to do. While preparing the data, we will spilt them into train and test with a ration of 75-25. We will also implement data cleaning and manipulation for modeling.

Data Visualization: We will show the data using K means clustering algorithm for players to find their best way of scoring and which team will win based on data clustering.

In this paper, the author used models like Linear Regression, Gradient Boosting Machine, Support Vector Machine, Neural network, Train Valid data results for Regression Analysis and classification analysis to find the results. These are the things that we studied in this class. But we will use Feature Weighting to predict who will win in next game.

III. Implementation

After reviewing some articles related to machine learning sports prediction, we were able to determine a method of implementation for predicting college basketball games. The implementation of our program consists of four main parts: importing the data, normalizing the data, training using feature weighting, and testing using feature weighting.

Step 1: Data Importation

The program utilizes 3 main data sets for training, and 2 auxiliary sets for testing. The training set are: one for player data [1], one for team data [2], and one for match data [3]. To keep things simple, the program only predicts matches for the 2018-2019 season. This season was chosen because it is the most recent season that also has an overlap of years with the other data sets. The match history covered 2018-2019, and the teams data set covers 2013-2019. The auxiliary testing sets are: one we created by referencing the results of the 2019 NCAA Men's Basketball tournament [7], and the other contains the predictions made by the program for the the same tournament.

We also removed features from the player and team data sets that were either incomplete or irrelevant. After all this, the player data set ended up containing 4,740 player entries, each with the player name, their team name, and 44 features. The team data set ended up with 353 team entries, each with the team name and 19 features. Finally, the match data set ended up with 5,463 match entries, each containing the date, the names of the two teams that played each other, and the two teams' final scores.

Before training begins, the program iterates through the teams data set to create a list with all unique team names. Then it iterates through the players data set to create a list of the players corresponding to those teams.

```
teams = []
players_in_teams = []
for i in range(len(team_data)):
   if not team_data[i][0] in teams:
      teams.append(team_data[i][0])
      players_in_teams.append([])
num_teams = len(teams)
teams.sort()
for i in range(int(len(player_data))):
   player_data[i][0] = player_data[i][0]

for i in range(len(player_data)):
   if player_data[i][1] in teams:
      players_in_teams[teams.index
        (player_data[i][1])].append
        (player_data[i][0])
```

This is important because we need to know which player is on which team so that we can not only use the team features for prediction, but also the player features.

We also find the maximum and minimum feature values for the player and team features. This will be important in the next step.

Step 2: Data Normalization

Next we use data normalization on each feature value in the player and team data sets. This is necessary in order to prevent features with higher values from having greater effects on the prediction. The formula used is the same one from the first article in the Literature Review section [4]:

$$X'_{i,t} = \frac{X_{i,t} - minX_i}{maxX_i - minX_i} \tag{17}$$

where i is the feature and t is the data point.

Step 3: Feature Weighting Training

Next, feature weighting is used to train and test the program. It works by utilizing a list of weights. This list contains a number value for each team feature (19 total). Initially, every feature has the same weight value, and during training, the program analyzes the features to determine which weights should increase and which should decrease. Features that contribute more to victory are given higher values for their weights and vice versa. This is determined by finding the average number of wins and the average feature values beforehand. The program uses the first half of the matches data set to determine the average number of wins (and the second half for testing).

```
for i in range(0, len(team_data)):
   if team_data[i][0] in match_teams:
      win_difference = match_team_wins[
          match_teams.index(team_data[i][0])]
          - average_wins
      for j in range(1, len(team_weights) + 1)
          :
         feature_difference = (team_data[i][j]
              - team_feature_means[j - 1])
         team_weights[j - 1] += (
             win_difference / average_wins) *
             (feature_difference /
             team_feature_means[j - 1])

for i in range(0, len(player_data)):
   if player_data[i][1] in match_teams:
      team_ind = -1
      for i in range(len(teams)):
         if teams[i] == player_data[i][1]:
            team_ind = i
            i = len(teams)
      if team_ind >= 0:
         win_difference = (match_team_wins[
             match_teams.index(player_data[i
             ][1])] - average_wins) / len(
             players_in_teams[team_ind])
         for j in range(2, (len(player_weights
             )) + 2):
            feature_difference = (player_data[
                i][j] - player_feature_means[j
                 - 2])
            player_weights[j - 2] += (
                win_difference / average_wins)
                 * (feature_difference /
                player_feature_means[j - 2])
```

Step 4: Feature Weighting Testing

Two types of testing were actually conducted. The first was on the regular season data. The second was on the NCAA tournament data, where we tested our predictions made by the trained algorithm for the tournament.

During the first set of testing, each weight is multiplied by its corresponding feature for a team, and the products are added together. Then it does the same for the player weights. The player values are divided by the number of players on the team so that teams with higher player counts are not given an advantage. The values are added up, and the team with the higher total value is predicted to win. The program compares the predicted winners with the actual winners to determine the accuracy.

```
num_correct = 0
num_total = 0
missing = 0
for i in range(int(len(match_data) / 2),
  len(match_data)):
   team1_val = 0
   team2_val = 0
   ind1 = -1
   ind2 = -1
   for k in range(0, len(team_data)):
      if team_data[k][0] == match_data
        [i][1]:
         ind1 = k
         if (ind1 >= 0) and (ind2 >= 0):
            k = len(team_data)
      elif team_data[k][0] ==
        match_data[i][3]:
         ind2 = k
         if (ind1 >= 0) and (ind2 >= 0):
            k = len(team_data)
   if (ind1 >= 0) and (ind2 >= 0):
      num_total += 1
      for j in range(1, len(team_data[0])):
         team1_val += (team_data[ind1][j] *
             team_weights[j - 1])
         team2_val += (team_data[ind2][j] *
             team_weights[j - 1])

   if match_data[i][1] in teams:
      team_ind = teams.index(match_data[i][1])
      player_ind = -1
      for k in range(len(player_data)):
         if player_data[k][1] == match_data[i
             ][1]:
            player_ind = k
      if player_ind >= 0 and team_ind >= 0:
         for j in range(2, (len(player_weights
             )
           ) + 2):
            team1_val += (player_data[
              player_ind][j] * player_
              weights[j - 2]) / len(
                  players_in_teams[team_ind])

   if match_data[i][3] in teams:
```

```
        team_ind = teams.index(match_data[i][3])
        player_ind = -1
        for k in range(len(player_data)):
            if player_data[k][1] == match_data[i
                ][1]:
                player_ind = k
        if player_ind >= 0 and team_ind >= 0:
            for j in range(2, (len(player_weights
                )) + 2):
                team2_val += (player_data[
                    player_ind][j] *
                    player_weights[j - 2]) / len(
                    players_in_teams[team_ind])

    if (team1_val > team2_val) and (match_data[
        i][2] > match_data[i][4]):
        num_correct += 1
    elif (team1_val < team2_val) and (
        match_data[i][2] < match_data[i][4]):
        num_correct += 1
print('Accuracy: ' + str(num_correct /
    num_total))
```

The second set was conducted in a very similar manner to the first set. However, the second set was on a considerably smaller data set. It consisted only of the 64 teams that were entered into the NCAA Men's basketball tournament. Like the first time, both teams were assigned a value, and the team with the higher value was considered the winner. But this time normalization was applied the the values given to each team. So instead of an arbitrary number, each team received a probability of winning. This process was used to fill out a tournament style bracket for the 64 teams. The predictions were made in rounds, similar to how the actual tournament would play out. The first round 32 predictions were made. Then, in the next round, the teams that moved on were made predictions for again. So the second round 16 predictions were made. This repeated until bracket was completely filled out. Lastly, to measure the accuracy, the actual tournament results were compared to the predictions.

IV. Conclusion

Based on our research of machine learning sports prediction, we were able to gain a better understanding of the problem and determine a sensible approach to solving it. We chose feature weighting for training and testing because the player and team data sets contained high numbers of features, allowing for improved accuracy using feature weighting. After everything was finished, we ran the program and ended up with a regular season accuracy of about 72.33%. The NCAA tournament predictions ended up at 44 out of 63 winners correctly predicted. This equates to roughly 69.84%. In the future, these results could potentially be further improved with higher data set dimensionality and slight adjustments to the algorithm.

References

[1] Kumar, A. 2021, "College Basketball 2009-2021 + NBA Advanced Stats", Verson 5. Retrieved, April 2022 from https://www.kaggle.com/datasets/adityak2003/college-basketball-players-20092021 [Accessed: Apr. 13, 2022].

[2] Sundberg, A. 2021. "College Basketball Dataset" Available: https://www.kaggle.com/datasets/andrewsundberg/college-basketball-dataset?resource=download [Accessed: Apr. 13, 2022].

[3] Massey, Kenneth. (2020, March). NCAA Regular Season Results, Version 1. https://www.kaggle.com/competitions/google-cloud-ncaa-march-madness-2020-division-1-mens-tournament/data [Accessed: May 1, 2022]

[4] Chi-Jie Lu, Tian-Shyug Lee, Chien-Chih Wang, and Wei-Jen Chen, "Improving Sports Outcome Prediction Process Using Integrating Adaptive Weighted Features and Machine Learning Techniques Processes"; Basel Vol. 9, Iss. 9, (2021) [Accessed: Apr. 13, 2022].

[5] Ji, Hao, et al. "March madness prediction: A matrix completion approach." Proceedings of Modeling, Simulation, and Visualization Student Capstone Conference. 2015 [Accessed: Apr. 13, 2022].

[6] Nguyen Hoang Nguyen, Duy Thien An Nguyen, Bingkun Ma Jiang Hu (2021) "The application of machine learning and deep learning in sport: predicting NBA players' performance and popularity, Journal of Information and Telecommunication", DOI: 10.1080/24751839.2021.1977066 [Accessed: Apr. 13, 2022].

[7] NCAA Tournament Bracket 2019. (n.d.). ESPN. http://www.espn.com/mens-college-basketball/tournament/bracket/_/id/201922/2019-ncaa-tournament [Accessed: May 1, 2022]