Problem Chosen	2021 MCM/ICM Summary Sheet	Team Control Number
A		46

Optimal solution to the 2020-2021 season NBA schedule

Summary

Due to the impact of the COVID-19 epidemic, the 2020-2021 season will be one of the more difficult seasons for the NBA. For TV ratings and commercial profits, the NBA hopes to hold as many games as possible without affecting the player's status. The player's status is directly related to the team's winning percentage. In this article, we discussed the impact of game density, number of consecutive away games, All-Star Game and other factors on winning percentage. Established a mathematical model based on the above analysis, and based on this model, formulated a game plan that minimizes back-to-back games.

First, we discussed the impact of game density and home and away factors on players. We analyzed the data of the 2018-2019 and 2019-2020 seasons, quantified the density of matches as game intervals, and calculated the number of consecutive away games. After that, we used a neural network model to predict the winner of each game based on four variables in the data: the number of consecutive away games, the interval between matches, and the strength of the two teams (related to the winning rate). And use this model to analyze the impact of match density (including back-to-back matches) on winning percentage.

Because the NBA has the tradition of the All-Star Game, the All-Star Game is also one of the most watched events in the NBA, which makes the timing of the All-Star Game particularly important. In this part, we established a statistical model to analyze the impact of the All-Star Game schedule on the regular season before and after the All-Star Game through hypothesis testing. And use the rules derived from this model to guide the timing of the All-Star Game.

In the formulation of specific game plans for the 2020-2021 season. Based on the previous discussion, we need to average the game interval as much as possible and reduce the number of back-to-back games to ensure the physical condition of the players and improve the quality of the game. This is a typical optimization problem. We used 0-1 planning model to solve the game plan with the fewest back-to-back games.

Our model can better analyze the game data of the 2018-2019 and 2019-2020 seasons, and get results that are consistent with general experience. At the same time, it also gives a game plan that minimizes back-to-back games. However, due to the fact that there are many factors that affect the results of the actual game, the deviation of the prediction result of our neural network model is too large, which needs to be further corrected. In general, this model takes into account the game interval, the number of away games and other factors, can give a game plan that minimizes the number of back-to-back games, and has certain guiding significance in the actual NBA schedule.

Keywords: Neural Network, Hypothesis Testing, 0-1 Programming

Contents

1 Introduction

1.1 Background

Affected by COVID-19, the 2020-2021 season will be a tougher season for the NBA.Due to the pandemic, it is obvious that the new 2020-2021 NBA seas on could not be started normally like the last season. For the sake of TV ratings and commercial profits, the NBA hopes to arrange as many games as possible within a limited time. However, a long and dense NBA schedule, especially the number of back to back games and consecutive away games, would greatly increase the risk of players injuries. Therefore, for the vote on the new season, NBPA must balance the players health and TV revenue.This puts new requirements on the NBA game schedule

1.2 Problem Statement

1. According to the schedules and results of 2018-2019 and 2019-2020 seasons,please establish a mathematical model to analyze the effects of the density of games and the number of back to back games on team winning. Then, please list the top three teams that were heavily affected by back to back games.In this question, we need to analyze the impact of match density on winning rate

2. By the schedule and results of 2018-2019 regular season, please establish a mathematical model to analyze the role of the All Star week for the subsequent games.Since the All-Star Game is also one of the most watched events in the NBA, its timing is particularly important. In this part, we need to analyze the impact of the All-Star Game schedule on the regular season before and after the All-Star Game.

3. Establish a model to design a regular game schedule that will be ended before April 15, 2021. In your schedule, you must ensure that each team plays the same number of home and away games while playing as few back to back games as possible.This requires us to optimize the game time arrangement.

2 Assumptions and Symbols

2.1 Model Hypothesis

1. During the regular season, the relative strength of each team remains unchanged.

2. It is assumed that the competition conditions in each venue are the same. That is, in different games, the away game has the same effect

3. It is considered that back-to-back matches are the case where the match interval is 0 days, that is, there are matches on two consecutive days.

4. It is assumed that there is no sudden suspension of the game caused by possible factors such as the intensification of the epidemic.

2.2 Symbols and Definitions

Table 1: Notation

Symbols	Definitions
SV_i	The strength value of the i-th team
α_i	The win rate of the i-th team
x_1	Strength of the team
x_2	The Strength of the opponent
x_3	The interval between of games
x_4	The number of consecutive away games
X_i	The i-th data in the dataset.

P.S. Other symbols instruction will be given in the text.

3 Neural Network Model

We first established a neural network model to illustrate the impact of game density and consecutive away games on the winning percentage of NBA teams.

3.1 The Features of Input Vector

Generally, neural network needs to have input vector. We denote it as:

$$X = \begin{bmatrix} x_1, & x_2, & x_3 & \cdots & x_n \end{bmatrix} \tag{1}$$

Where: x_i is the i-th feature in the input vector.

The features in the input vector represent the factors that affect the team's winning rate. There are many factors that affect a teams winning percentage. In the end, we selected four features that have the greatest impact on the team's victory, namely **the strength of the team**, **the strength of the opponent**, **the density of the team's game**, and **the number of consecutive away games**.

3.1.1 the strength of the team

Team strength is the most important factor in determining the team's victory. But the strength of the team is not a specific value, which brings trouble to the analysis of the problem. In order to facilitate the analysis of the problem, we introduced the concept of **team strength value** and quantified the team's strength into numerical values. Obviously, the team's strength value is positively correlated with the team's winning percentage. It can be considered that the higher the team's winning rate, the greater the team's strength value. Therefore, for the sake of simplicity, we use the teams average winning percentage in a season as the teams strength value and normalize the results. Because the 19-20 season was affected by the epidemic, we used the 18-19 season data as a reference. We use statistical methods to get the winning percentage of each team. The specific strength values of each team are presented in the table below.

Table 2: The winning percentage and strength value of 30 teams.

team	win rate	SV	team	win rate	SV
Milwaukee Bucks	0.732	3.536	Detroit Pistons	0.5	2.415
Toronto Raptors	0.707	3.415	Charlotte Hornets	0.476	2.300
Golden State Warriors	0.695	3.357	Miami Heat	0.476	2.300
Denver Nuggets	0.659	3.184	Sacramento Kings	0.476	2.300
Portland Trail Blazers	0.646	3.121	Los Angeles Lakers	0.451	2.179
Houston Rockets	0.646	3.121	Minnesota Timberwolves	0.439	2.121
Philadelphia 76ers	0.622	3.005	Memphis Grizzlies	0.402	1.942
Utah Jazz	0.610	2.947	New Orleans Pelicans	0.402	1.942
Boston Celtics	0.598	2.889	Dallas Mavericks	0.402	1.942
Oklahoma City Thunder	0.598	2.889	Washington Wizards	0.39	1.884
Indiana Pacers	0.585	2.826	Atlanta Hawks	0.354	1.710
San Antonio Spurs	0.585	2.826	Chicago Bulls	0.268	1.295
Los Angeles Clippers	0.585	2.826	Cleveland Cavaliers	0.232	1.121
Brooklyn Nets	0.512	2.473	Phoenix Suns	0.232	1.121
Orlando Magic	0.512	2.473	New York Knicks	0.207	1.000

In order to visually display the data in the above table, we make the following two pictures:

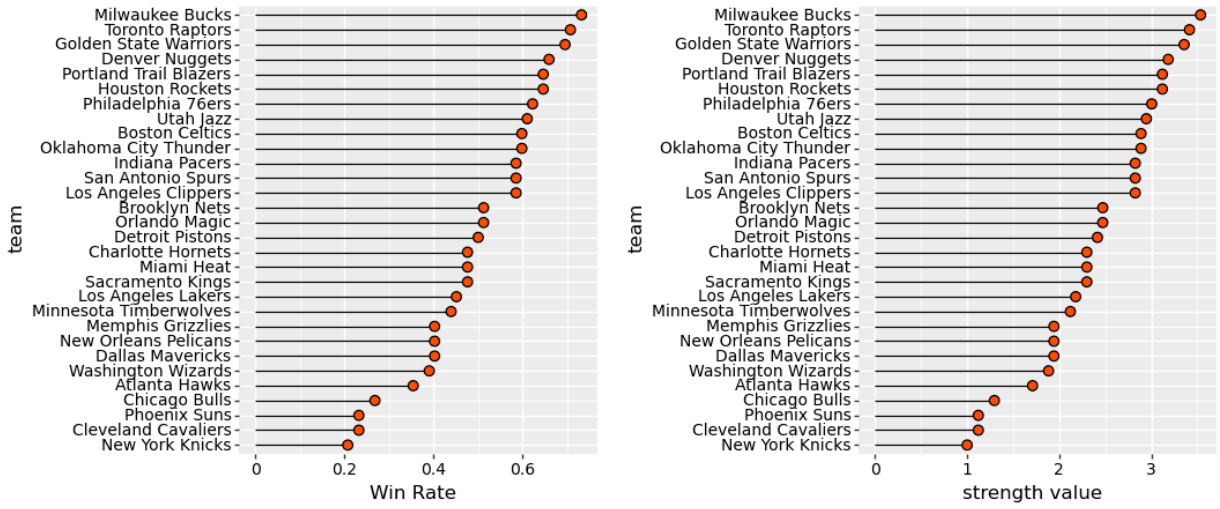

Figure 1: The picture on the left represents the winning percentage of each team, and the picture on the right represents the strength value of each team. We assume that the team with the lowest winning percentage has a strength value of 1.

The strength of the team is the first element of the input vector of our neural network model, and we denote it as x_1.

3.1.2 the strength of the opponent

The strength of the opponent is also an important factor affecting the victory of the team. If the opponent's strength is weak, the chances of the team winning will increase. So our model also considers the opponent's strength value. The strength of the opponent is the second element of the input vector of our neural network model, and we denote it as x_2.

3.1.3 the density of games

The density of the team's game also affects the team's winning percentage. Teams experiencing highly competitive densities may be particularly exposed to performance breakdown and injury risk. [1] Appropriate game density is essential to the team's performance. In order to study the relationship between the density of the game and the winning percentage of the team, we divide the density of the game into four levels:

- **strong**
  There are games for two or more consecutive days, which is what we usually call back to back.
- **average**
  There is one day off during the game, and there is a day interval between games.
- **weak**
  There are two days off during the game, and there is a two-day interval between games.
- **particularly weak**
  There are three days or more between two games.

In our data set, we use 1, 2, 3, 4 to represent the four densities of the game. The density of games is the third element of the input vector of our neural network model, and we denote it as x_3.

3.1.4 consecutive away games

The playing field has a great influence on the team's winning percentage. The teams at home games are more familiar with the environment and generally have better performance. The away game teams often fail to perform at their best due to exhaustion from the journey and environmental discomfort. [2]Therefore, away games are not good for the team, and consecutive away games cause the team to travel back and forth between different cities, which is even more unfavorable for the team. In order to analyze the influence of consecutive away games on the team's winning rate, we take **the number of consecutive away games** as the fourth element of the input vector, which is denoted as x_4.

3.2 Output Vector

Obviously, the output vector is a scalar with a dimension of 1×1, indicating whether the team wins or loses. We use 1 to indicate that the team wins and 0 to indicate that the team loses.

3.3 Build Neural Network

Artificial neural network is an algorithmic mathematical model that imitates the behavioral characteristics of animal neural networks and performs distributed and parallel information processing. This kind of network relies on the complexity of the system and achieves the purpose of processing information by adjusting the interconnection between a large number of internal nodes.

3.3.1 the structure of neural networks

After comparative analysis, we choose to build a neural network with a hidden layer and four nodes in the hidden layer. The following figure is a schematic diagram of the neural network structure.

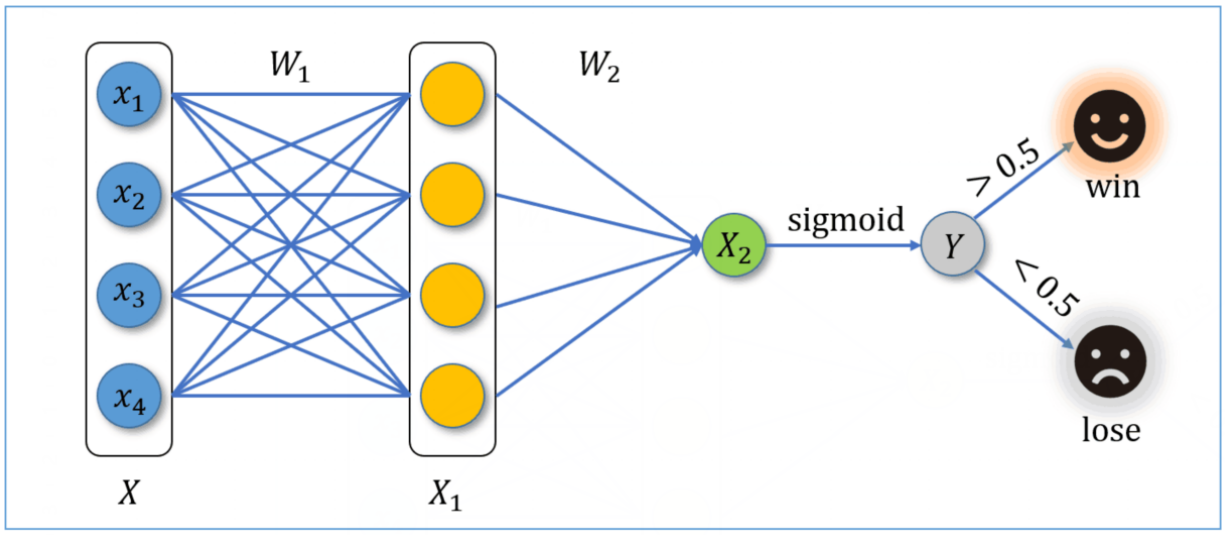

Figure 2: The structure of our neural network.

3.3.2 Neural network training algorithm

We use the sigmoid function: $sigmoid(x) = \frac{1}{1+e^{-x}}$ as the activation function of the neuron. In the training set $D = (X_1, y_1), (X_2, y_2), \cdots, (X_i, y_i)$, suppose the prediction result of the neural network is $\hat{y_k} = (\hat{y_1^k}, \hat{y_2^k}, \cdots, \hat{y_l^k})$, then the mean square error of prediction is:

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(\hat{y_j^k} - y_j^k)^2 \tag{2}$$

In the solution of the single hidden layer feedforward network shown in the figure above, our goal is to minimize the cumulative mean square error $E = \frac{1}{m}\sum_{k=1}^{m} E_k$.We used the error back propagation algorithm [3] based on gradient descent strategy for solving the connection rights of the neural network w_{hj}, w_{ih} and thresholds θ_j, γ_h .

Suppose the input of the h-th neuron in the hidden layer is $\alpha_h = \sum_{h=1}^{q} v_{ih} x_i$, and the input of the j-th neuron in the output layer is $\beta_j = \sum_{h=1}^{q} w_{hj} b_h$, we can conclude:

For the error number calculated in the formula 2, under the learning rate η, there is:

$$\Delta W_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} \tag{3}$$

Note that $w_h j$ first affects the input β_j of the j-th input layer, then affects its output $\hat{y}_j^k$, and then affects E_k, there are:

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \frac{\partial \hat{y}_j^k}{\partial \beta_j} \frac{\partial \beta_j}{\partial w_{hj}} \tag{4}$$

Let b_h represent the output of the h-th neuron in the hidden layer,Simplify the above formula to get:

$$b_h = \frac{\partial \beta_j}{\partial w_{hj}} \tag{5}$$

According to the definition of neurons and the input and output of the model, there are:

$$g_j = \hat{y}_j^k (1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k) \tag{6}$$

The update formula of the available parameters by combining two formulas 5 and 6($e_h = b_h(1 - b_h) \sum_{j=1}^{l} w_{hj} g_j$):

$$\begin{cases} \delta w_{hj} = \eta g_j b_h \\ \delta \theta_j = \eta g_j \\ \delta v_i h = \eta e_h x_i \\ \delta \gamma_h = -\eta e_h \end{cases} \tag{7}$$

The summarized BP algorithm is summarized as follows:

Algorithm 1: Back Propagation Algorithm

Input: Dataset: $D = (X_i, y_i)_{k=1}^{m}$; Learning rate η;

Output: Multi-layer feedforward neural network with connection weight and threshold determined.

```
Initialize all connection weights and thresholds of the network in the range of (0, 1);
repeat
    forall (X_k, y_k) ∈ D do
        Calculate the sample output ŷ_k according to the current parameters;
        Calculate the gradient term g_j of the output layer as 6
        Calculate the gradient term g_h of the hidden layer
        Update connection rights w_hj, w_ih and thresholds θ_j, γ_h as 7
until Stop condition reached;
```

3.3.3 training and testing

Neural networks need a lot of data for training, and the remaining part is used for testing. We collected a total of 2,460 games in the 18-19 season, and separately counted the strength of the participating teams, the strength of the opponent, the density of the game, the number of consecutive away games, and the wins and losses. The results are recorded in the attached excel table. We chose 2000 data as the training set and 120 data as the test set. We set the learning rate to 0.01.

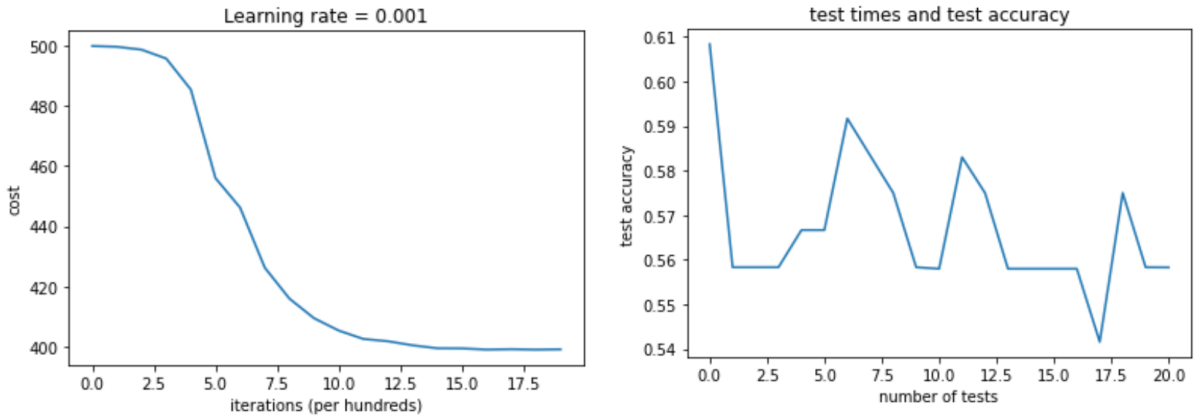

Figure 3: The picture on the left shows the change of the loss value with the number of iterations when training the model. It can be seen that when the number of iterations is greater than 1500, the loss value basically does not decrease, indicating that the model training is over. The picture on the right shows the accuracy on the test set. We trained and tested it 20 times.

3.4 solving and analysis

In this part, we use the trained neural network model to analyze the relationship between game density, number of consecutive away games, and team winning percentage.

3.4.1 match density and winning rate

In order to study the relationship between match density and team win rate, we first control other factors unchanged and only change the team's match density. We use the average strength value of 30 teams to replace the strength value of the team and the opponent, and assume that the team is playing at home. We can get the input vector:

$$X = \begin{bmatrix} x_1, & x_2, & x_3, & x_4 \end{bmatrix} \tag{8}$$

Where: $x_1 = x_2 = \bar{SV}$, $x_3 = 1, 2, 3, 4 \cdots$, which indicates the density of the game, and $x_4 = 0$, indicating that the number of consecutive away games is 0, that is, at home games.

We can get:

$$\bar{SV} = \frac{1}{30} \times \sum_{i=1}^{30} SV_i \tag{9}$$

We compute it as 2.415333 and made the image on the left of **Fig.4**.

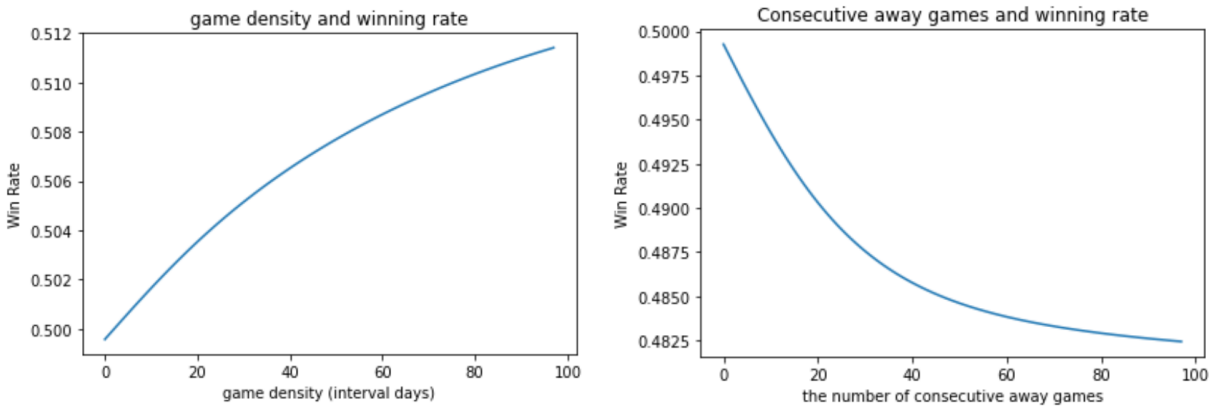

Figure 4: In the left picture, the horizontal axis represents the number of days between two matches, and the vertical axis represents the winning percentage. In the right picture, the horizontal axis represents the number of consecutive away games and the vertical axis represents the winning percentage.

We can see that as the density of matches decreases (the number of days between matches increases), the team's winning percentage is gradually increasing. Assuming the strength of the two teams is equal, then the home team has a higher winning percentage, which is in line with common sense.

3.4.2 the number of consecutive away games and winning rate

To study the relationship between consecutive away games and team winning percentage, we only need to change x_4, which indicates the number of consecutive away games. We set x_3 to 2, that is, one day between the two games. Other variables are the same as in the previous section. Finally we get the picture on the right in **Fig.4**. We can see that as the number of consecutive away games increases, the team's winning percentage gradually weakens. The reason for this situation may be fatigue caused by the journey and inadaptability to the environment.

3.5 Back-to-back games and winning rate

In this part, we calculated the back-to-back winning percentage and the non-back-to-back winning percentage of the 30 teams in 2018-2019 season3 to determine which team is most affected by back-to-back gaming.

Table 3: The winning percentage of NBA teams in back-to-back and non-back-to-back situations

Team	B-B WR	Non-B-B	delta	Team	B-B WR	Non-B-B	delta
Wizards	0.667	0.328	-0.338	Bulls	0.357	0.25	-0.107
Spurs	0.308	0.638	0.33	Lakers	0.539	0.435	-0.104
Nuggets	0.923	0.609	-0.314	Clippers	0.5	0.603	0.103
Mavericks	0.143	0.456	0.313	Magic	0.429	0.529	0.101
76ers	0.385	0.667	0.282	Knicks	0.133	0.224	0.091
Nets	0.286	0.559	0.273	Hawks	0.417	0.343	-0.074
Trail Blazers	0.417	0.686	0.269	Pacers	0.643	0.574	-0.069
Celtics	0.385	0.638	0.253	Raptors	0.75	0.7	-0.05
Timberwolves	0.231	0.478	0.248	Pistons	0.462	0.507	0.046
Bucks	0.539	0.768	0.23	Heat	0.5	0.471	-0.029
Kings	0.286	0.515	0.229	Rockets	0.667	0.643	-0.024
Grizzlies	0.231	0.435	0.204	Suns	0.25	0.229	-0.021
Pelicans	0.231	0.435	0.204	Cavaliers	0.214	0.235	0.021
Hornets	0.333	0.508	0.174	Thunder	0.583	0.6	0.017
Jazz	0.5	0.632	0.132	Warriors	0.692	0.696	0.003

From the absolute value analysis of the change in winning rate, **the Washington Wizards, San Antonio Spurs and Denver Nuggets** are the three teams most affected by back-to-back. However, the back-to-back winning percentage of Washington Wizards and Denver Nuggets is higher than non-back-to-back ones.This may be related to the fact that these three teams happened to encounter weaker opponents in back-to-back matches.

Excluding the situation where the back-to-back winning rate is higher, the three teams most affected by back-to-back matches are **San Antonio Spurs, Dallas Mavericks and Philadelphia 76ers**.

4 Statistical model and the All-Star Game

4.1 Background information and data

After a long-term development, the contemporary NBA All-Star Game has developed into a 3-day "All-Star Week". The events of the All-Star Game include "All-Star Party", "Bell Skills Challenge" , "Slam Dunk" ,"All-Star Game" ,etc. All-Star Week is usually between the first half and the second half of the regular season.

In our model, we analyzed 2,290 games in the 2018-2019 season and the 2019-2020 regular season to reveal the impact of the All-Star Game on winning percentage. The total winning percentage, the pre-All-Star winning percentage and the post-All-Star winning percentage of the 30 participating teams are respectively drawn as the chart 4.1 follows: And compare the three winning percentages of each team to get the influence of the All-Star Game on the regular season. Since there are less than 1230 games in the regular season in the 2019-2020 season, the following data is mainly based on the 2018-2019 season.

Table 4: The winning percentage of NBA teams in the 2018-2019 season(WP refers to Winning Percentage)

Team	Pre WP	Post WP	Team	Pre WP	Post WP
Milwaukee Bucks	0.7543	0.6800	Detroit Pistons	0.4642	0.5769
Toronto Raptors	0.7288	0.6522	Charlotte Hornets	0.4736	0.4800
Golden State Warriors	0.7192	0.6400	Miami Heat	0.4642	0.5000
Denver Nuggets	0.6842	0.6000	Sacramento Kings	0.5263	0.3600
Houston Rockets	0.5789	0.8000	Los Angeles Lakers	0.4912	0.3600
Portland Trail Blazers	0.5964	0.7600	Minnesota Timberwolves	0.4736	0.3600
Philadelphia 76ers	0.6379	0.5833	Memphis Grizzlies	0.3898	0.4348
Utah Jazz	0.5614	0.7200	New Orleans Pelicans	0.4406	0.3043
Boston Celtics	0.6379	0.5000	Dallas Mavericks	0.4561	0.2800
Oklahoma City Thunder	0.6491	0.4800	Washington Wizards	0.4137	0.3333
Indiana Pacers	0.6551	0.4167	Atlanta Hawks	0.3275	0.4167
San Antonio Spurs	0.5593	0.6522	Chicago Bulls	0.2413	0.3333
Los Angeles Clippers	0.5423	0.6957	Cleveland Cavaliers	0.2068	0.2917
Brooklyn Nets	0.5084	0.5217	Phoenix Suns	0.1864	0.3478
Orlando Magic	0.4576	0.6521	New York Knicks	0.1896	0.2500

4.2 Hypothesis testing model

In the subsequent analysis, we ranked the teams according to their winning percentage and divided the teams from strong to weak into three groups, A, B, and C, with ten teams in each group. According to the mean and variance of the winning percentage of each group, we can infer the impact of the All-Star Game on teams of different levels. Regarding this effect, we propose the hypothesis: the All-Star Game has no effect on the team's performance, that is, the team's winning rate before the All-Star game μ_{pre} is equal to the winning rate after the game μ_{post}. That is, suppose $H_0 : \mu_{pre} = \mu_{post}$. Based on the sample data, select the statistic 10 to perform hypothesis testing.

$$Z = \frac{\bar{X} - \bar{Y} - 0}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \tag{10}$$

From the above formula, we know that for group A ,$z_a = 3.9722$, for group B ,$z_b = -1.6876$, and for group C ,$z_c = -0.6630$. From the standard normal distribution and hypothesis test, it can be obtained: For group A, there is a 99.9% certainty to negate H_0, adopt $H_1 : \mu_{pre} \neq \mu_{post}$, for group B there is 90% We must deny H_0 and adopt $H_1 : \mu_{pre} \neq \mu_{post}$. For group C, consider H_0 established. Further modify the hypothesis, according to the one-sided hypothesis test, we can get 99.9% certainty to negate $H_0 : \mu_{pre} \leqslant \mu_{post}$ and adopt $H_1 : \mu_{pre} > \mu_{post}$, 95% sure to deny $H_0 : \mu_{pre} \geqslant \mu_{post}$ for Group B, and adopt $H_1 : \mu_{pre} < \mu_{post}$.

Based on the above discussion, it can be considered that in most cases, the All-Star Game will reduce the winning rate of the best-performing group A, and increase the winning rate of the medium-performing group B, but will have little effect on the worst-performing group C. This can

be explained as: some outstanding players in Group A need to prepare for the All-Star Game or lack of rest after participating in the All-Star Game, and their performance level decreases, which reduces the winning rate of Group A and increases the winning rate of Group B with certain strength. Due to its poor strength, Group C is less affected by the fluctuations in the level of players in Group A, and its winning rate is not greatly affected by the All-Star Game.

5 Game Schedule Model

The NBA is divided into the Eastern Conference and the Western Conference. Each league is divided into three divisions, each of which consists of five teams. The following table is a specific division and the number of each team:

Eastern Conference					
Atlantic Division		Central Division		Southeast Division	
1	Philadelphia 76ers	6	Milwaukee Bucks	11	Atlanta Hawks
2	Brooklyn Nets	7	Indiana Pacers	12	Charlotte Hornets
3	Boston Celtics	8	Cleveland Cavaliers	13	Orlando Magic
4	New York Knicks	9	Chicago Bulls	14	Miami Heat
5	Toronto Raptors	10	Detroit Pistons	15	Washington Wizards

Western Conference					
Northwest Division		Pacific Division		Southwest Division	
16	Utah Jazz	21	Los Angeles Clippers	26	San Antonio Spurs
17	Denver Nuggets	22	Los Angeles Lakers	27	Memphis Grizzlies
18	Portland Trail Blazers	23	Phoenix Suns	28	Houston Rockets
19	Oklahoma City Thunder	24	Golden State Warriors	29	Dallas Mavericks
20	Minnesota Timberwolves	25	Sacramento Kings	30	New Orleans Pelicans

NBA game scheduling is a difficult topic. To arrange an appropriate game schedule, many factors need to be considered. The game schedule of each team must first meet the following basic conditions:

1. 4 games with ones of the same division (2 for the home court and 2 for guest).
2. 4 or 3 games with teams of differ divisions but in the same conference (2 home and 1 guest, or 2 guest and 1 home). To be specific, 4 games will be played against 6 of the 10 teams and 3 games will be played against the remaining 4 teams.
3. 2 games with ones of another conference (1 for home and 1 for guest).

The NBA's partition situation is shown in the following table, where A is used to refer Atlantic Division, C is used to refer Central Division, P is used to refer Pacific Division, SE is used to refer Southeast Division,NW is used to refer Northwest Division and SW is used to refer Southwest Division, the game situation is as follows:

A vs A	C vs A	SE vs A	NW vs A	P vs A	SW vs A
A vs C	C vs C	SE vs C	NW vs C	P vs C	SW vs C
A vs SE	C vs SE	SE vs SE	NW vs SE	P vs SE	SW vs SE
A vs NW	C vs NW	SE vs NW	NW vs NW	P vs NW	SW vs NW
A vs P	C vs P	SE vs P	NW vs P	P vs P	SW vs P
A vs SW	C vs SW	SE vs SW	NW vs SW	P vs SW	SW vs SW

Because back-to-back games will affect the performance of the team, it is necessary to meet the minimum number of back-to-back games in the game schedule. We understand back-to-back games as the teams two or more consecutive games are on the away field.

5.1 Build the Model

We calculated that there are 114 days from the start to the end of the 20-21 season. Considering that the All-Star game lasts for 7 days, we need to make a 107-day game schedule. We make a competition matrix for each day's competition, a total of 107, our purpose is to calculate these 107 matrices.

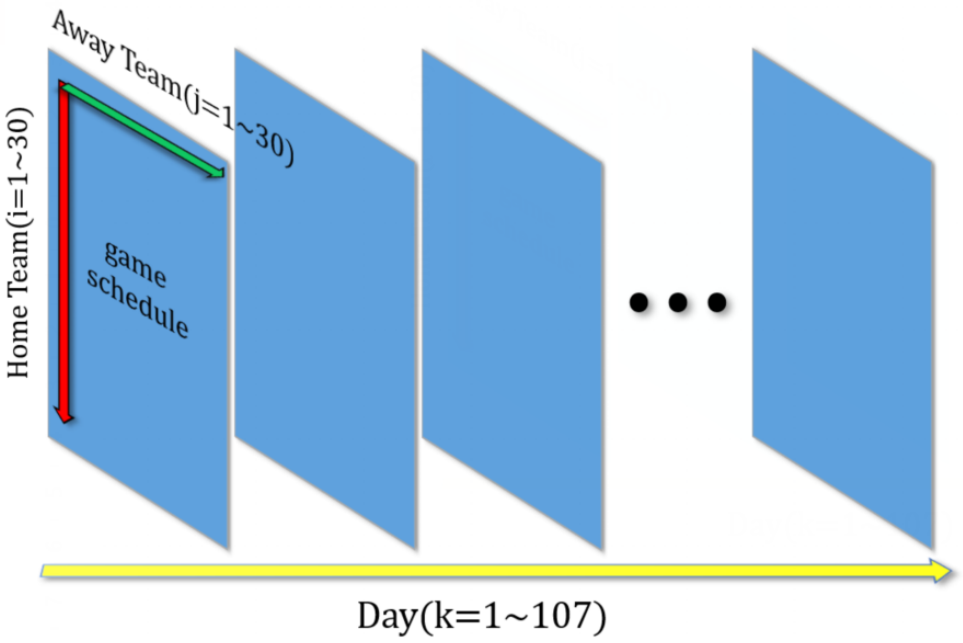

Figure 5: Schematic diagram of the competition schedule matrix.

The structure of our schedule matrix is shown in **Fig.5**. The rows of the matrix represent the 30 teams on the home field, and the columns of the matrix represent the 30 teams on the away field. We use a_{ijk} to represent the element in the i-th row and j-th column in the k-th matrix. When $a_{ijk} = 1$, it means there is a game between home team i and away team j on the k-th day. Otherwise, the two teams have no games on this day. We turn the problem into a mathematical programming problem.

We have the following constraints:

5.1.1 games between the same division

The following conditions must be met for games between the same division:

$$\sum_{k=1}^{107} a_{ijk} = 2 \tag{11}$$

where: $i \neq j$ and $1 \leq i, j \leq 5$ or $6 \leq i, j \leq 10$ or $11 \leq i, j \leq 15$ or $16 \leq i, j \leq 20$ or $21 \leq i, j \leq 25$ or $26 \leq i, j \leq 30$.

5.1.2 games between different divisions in the same conference

The following conditions must be met for games between different divisions in the same conference:

$$\sum_{k=1}^{107} a_{ijk} = 2 \quad or \quad 1 \tag{12}$$

where: $1 \leq i \leq 5$, $6 \leq j \leq 15$ or $6 \leq i \leq 10$, $1 \leq j \leq 5$, $11 \leq j \leq 15$ or $11 \leq i \leq 15$, $1 \leq j \leq 10$ or $16 \leq i \leq 20$, $21 \leq j \leq 30$ or $21 \leq i \leq 25$, $16 \leq j \leq 20$, $26 \leq j \leq 30$ or $26 \leq i \leq 30$, $16 \leq j \leq 25$.

Because there are three or four games between different divisions of the same conference, this situation is more complicated. We can be sure that out of 10 different $\sum_{k=1}^{107} a_{ijk}$ in one case, 8 of them are equal to 2, and the remaining are equal to 1. We can also determine that if $\sum_{k=1}^{107} a_{ijk}$ is equal to 1, then $\sum_{k=1}^{107} a_{jik}$ must be equal to 2. We manually adjust items in the game schedule based on these constraints.

5.1.3 games between different conference

The following conditions must be met for games between the different conference:

$$\sum_{k=1}^{107} a_{ijk} = 1 \tag{13}$$

where: $1 \leq i \leq 5$, $16 \leq j \leq 30$ or $6 \leq i \leq 10$, $16 \leq j \leq 30$ or $11 \leq i \leq 15$, $16 \leq j \leq 30$ or $16 \leq i \leq 20$, $1 \leq j \leq 15$ or $21 \leq i \leq 25$, $1 \leq j \leq 15$ or $26 \leq i \leq 30$, $1 \leq j \leq 15$.

5.1.4 other constraints

Finally, we must ensure that each team only plays one game a day:

$$\sum_{i=1}^{30} a_{ijk} + \sum_{j=1}^{30} a_{ijk} \leq 1, (i = j, 1 \leq i, j \leq 30) \tag{14}$$

5.1.5 objective function

We want to minimize the number of consecutive away games, so the following function is used as the objective function:

$$\max E = \sum_{k=1}^{106} \sum_{j=1}^{30} (|\sum_{i=1}^{30} a_{ij(k+1)} - \sum_{i=1}^{30} a_{ijk}|) \tag{15}$$

When team j has an away game on day k, $\sum_{i=1}^{30} a_{ijk} = 1$, whether it equals 0.

$|\sum_{i=1}^{30} a_{ij(k+1)} - \sum_{i=1}^{30} a_{ijk}| = 1$ means that there is an away game on one day and no away game(home game or no game) on the other day in two consecutive days. When we find the maximum value of this objective function and use it to arrange the game schedule, we can reduce the number of consecutive away games to a certain extent.

5.2 Solving the Model

We got a possible game schedule based on the above conditions and put it in the appendix.

6 Strengths and weaknesses

6.1 Strengths

- The model comprehensively takes into account the game interval, home and away games and other factors that affect the game that are easy to obtain and control, and have a guiding role in the formulation of the game plan.
- Comprehensive use of neural network models, statistical models, optimization models and other mathematical models, the model has strong adaptability to data and has a wide range of applications.

6.2 Weaknesses

- The prediction effect of the neural network model is poor, which may lead to a certain deviation between the model's estimate of the impact of the game interval on the winning rate and the actual situation.
- The model does not model the situation similar to the sudden cancellation of NBA games due to COVID19 in early 2020, and may have certain limitations in practical applications in 2021.

7 A Letter to NBA

Dear NBA

As one of the four major professional sports leagues in the United States, the NBA is the world leader in basketball. Due to the NBAs good organizational skills and the excellent level of the leagues players, NBA games are also one of the most exciting and spectacular basketball games in the world. In addition to the game, the strong and diligent spirit of NBA players has also inspired generation after generation to learn. For these reasons, the NBA league has become one of the most popular sports events in the world. It is also one of the most successful sports events that balances sportsmanship and commercialization.

The COVID-19 outbreak broke out in the middle of the 2019-2020 season, making the NBA game off for more than four months. Due to the league highly commercialized, 2020 had became one of the most difficult years in the history of the NBA league. As the current COVID-19 epidemic in the United States is still very serious, and the league has a certain profit demand, this orders certain requirements for the organization of the NBA league. How to arrange as many games as possible in a limited time without affecting the player's state has become a focus.

In order to arrange as many matches as possible without affecting the state of the team, we need to analyze the impact of match density on the state of the team. In order to facilitate analysis, here we quantify this indicator as the impact of game density on team strength (winning rate).First, we used a neural network model to analyze the impact of two important factors affecting the winning rate of the game, the number of consecutive away games and the number of days between games.And got a mathematical model that can predict the game situation based on the above two parameters and the teams average winning percentage.According to this model, the influence of game density and away situation on winning rate is quantified.

The All-Star Game is the most watched event in the NBA, so determining the timing of the All-Star Game is also the top priority of the NBA schedule. We use the mathematical method of hypothesis testing to determine the fact that the All-Star Game may have a certain negative impact on the stronger teams and help improve the performance of the mid-level teams. This helps to formulate the regular season schedule before and after the All-Star Game, and reasonably determine when the All-Star Game will be held.

In formulating the actual game plan for the 2020-2021 season, we started from all possible situations of the game plan, and followed the actual schedule of the game restrictions given by the previous model and actual event schedule like "No two matches of the same team on the same day" , to narrow down the feasible domain of determining the game plan. In order to protect the player's state as much as possible, we need to minimize the number of back-to-back matches in the schedule, which is a typical 0-1 planning problem. According to the knowledge of mathematics, we find the best competition plan as what is shown in"GameSchedule.xlsx"

As a loyal NBA spectator, I sincerely hope that the 2020-2021 NBA games can be played normally, and I also hope that the above content can be helpful to your game schedule.

Sincerely yours,
Your friends

References

[1] Pedro T Esteves, Kazimierz Mikolajec, Xavier Schelling, and Jaime Sampaio. Basketball performance is affected by the schedule congestion: Nba back-to-backs under the microscope. *European journal of sport science*, pages 1–10, 2020.

[2] DR Junior. Statistical analysis of basketball performance indicators according to home/away games and winning and losing teams. *Journal of Human Movement Studies*, 47:327–336, 2004.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Appendices

Appendix A Python code for data processing

This code implements a simple reading and processing of data. And calculated the winning percentage in different situations.

```
import xlrd
import xlwt
import re
from datetime import datetime, timedelta

input_file_name = '2018-2019.xlsx'
# 30
name_list = ["Philadelphia 76ers", "Boston Celtics", "Oklahoma City Thunder",
             "Golden State Warriors","Milwaukee Bucks", "Charlotte Hornets",
             "Brooklyn Nets", "Detroit Pistons", "Memphis Grizzlies",
             "Indiana Pacers", "Miami Heat", "Orlando Magic", "Atlanta Hawks",
             "New York Knicks","Cleveland Cavaliers", "Toronto Raptors",
             "New Orleans Pelicans", "Houston Rockets","Minnesota Timberwolves",
             "San Antonio Spurs", "Utah Jazz", "Sacramento Kings", "Denver Nuggets",
             "Los Angeles Clippers", "Dallas Mavericks", "Phoenix Suns",
             "Chicago Bulls", "Washington Wizards","Portland Trail Blazers", "Los Angeles La

def getgames(name_list, visitor, host,date):
    games = []
    games_before=[]
    games_after=[]
    for i in range(len(name_list)):
        games.append(0)
        games_before.append(0)
        games_after.append(0)
    for k in range(len(name_list)):
        for i in range(len(host)):
            if name_list[k] == visitor[i] or name_list[k] == host[i]:
                games[k] = games[k] + 1
```

```
                if date[i]-datetime(2019,2,17)<timedelta(days=0):
                    games_before[k]=games_before[k]+1
                else:
                    games_after[k]=games_after[k]+1
    return games,games_before,games_after


# excel
def read_excel(input_file_name):
    """
    xls
    """
    workbook = xlrd.open_workbook(input_file_name)
    print(workbook)
    # workbooksheet_names()excel
    print(workbook.sheet_names())
    # sheet_by_index()sheet_by_name()
    # table = workbook.sheet_by_index(0)
    # print(table)
    table = workbook.sheet_by_name('Sheet1')
    print(table)
    # nrowsncols
    rows = table.nrows
    cols = table.ncols
    #
    date = []
    visitor = []
    host = []
    winner = []
    for row in range(rows):
        for col in range(cols):
            data = table.cell(row, col).value
            if (data == "Date"):
                break
            if (col == 0):
                #
                data = data[5:]
                month = data[:3]
                if month == "Jan":
                    month = "01"
                elif month == "Feb":
                    month = "02"
                elif month == "Mar":
                    month = "03"
                elif month == "Apr":
                    month = "04"
                elif month == "Jul":
                    month = "07"
                elif month == "Aug":
                    month = "08"
                elif month == "Oct":
                    month = "10"
                elif month == "Nov":
                    month = "11"
```

```
                elif month == "Dec":
                    month = "12"
                day = re.findall(r"\d+", data[4:6])
                if (int(day[0]) < 10):
                    day[0] = "0" + day[0]
                year = re.findall(r"\d+", data[7:])
                data = year[0] + month + day[0]
                date.append(data)
            elif (col == 1):
                visitor.append(data)
            elif (col == 3):
                host.append(data)
            elif (col == 5):
                winner.append(data)
    return date, visitor, host, winner


#
def DateRegular(date):
    year = []
    month = []
    day = []
    d = []
    for i in range(len(date)):
        year.append(int(date[i][0:4]))
        month.append(int(date[i][4:6]))
        day.append(int(date[i][6:]))
        d.append(datetime(year[i], month[i], day[i]))
    return d


#
def SheetInit(date, visitor, host, winner):
    workbook = xlrd.open_workbook('DataSet2.xls')
    table = workbook.sheet_by_name('sheet')
    book = xlwt.Workbook(encoding="utf-8")
    worksheet = book.add_sheet('sheet', cell_overwrite_ok=True)
    worksheet.write(0, 0, "Date")
    worksheet.write(0, 1, "Opponent1")
    worksheet.write(0, 2, "Opponent2")
    worksheet.write(0, 3, "Winner")
    worksheet.write(0, 4, "Intervals")
    worksheet.write(0, 5, "AwayGames")
    worksheet.write(0, 6, "Power1")
    worksheet.write(0, 7, "Power2")
    worksheet.write(0, 8, "Prediction")
    #
    for i in range(len(date)):
        worksheet.write(2 * i + 1, 0, date[i])
        worksheet.write(2 * i + 2, 0, date[i])
        worksheet.write(2 * i + 1, 1, visitor[i])
        worksheet.write(2 * i + 2, 1, host[i])
        worksheet.write(2 * i + 1, 2, host[i])
        worksheet.write(2 * i + 2, 2, visitor[i])
```

```
        worksheet.write(2 * i + 1, 3, winner[i])
        worksheet.write(2 * i + 2, 3, winner[i])
        worksheet.write(2 * i + 2, 5, 0)
        worksheet.write(2 * i + 1, 6, CalcWin(visitor[i],[0],[0],winner, games)[0])
        worksheet.write(2 * i + 2, 6, CalcWin(host[i],[],[],winner, games)[0])
        worksheet.write(2 * i + 1, 7, CalcWin(host[i], [],[],winner, games)[0])
        worksheet.write(2 * i + 2, 7, CalcWin(visitor[i],[],[], winner, games)[0])
        if table.cell(2 * i + 1, 1).value == table.cell(2 * i + 1, 3).value:
                worksheet.write(2 * i + 1, 8, 1)
                worksheet.write(2 * i + 2, 8, 0)
        else:
                worksheet.write(2 * i + 1, 8, 0)
                worksheet.write(2 * i + 2, 8, 1)
        j = i - 1
        while j >= 0:
            if table.cell(2 * j + 2, 1).value == table.cell(2 * i + 2, 1).value:
                value = table.cell(2 * i + 2, 0).value - table.cell(2 * j + 2, 0).value
                if value >= 4: value = 4;
                worksheet.write(2 * i + 2, 4, value)
                worksheet.write(2 * i + 1, 4, value)
                break
            elif table.cell(2 * j + 1, 1).value == table.cell(2 * i + 1, 1).value:
                value = table.cell(2 * i + 1, 0).value - table.cell(2 * j + 1, 0).value
                if value >= 4: value = 4;
                worksheet.write(2 * i + 1, 4, value)
                worksheet.write(2 * i + 2, 4, value)
                break
            j = j - 1
        if (j <= 0):
            worksheet.write(2 * i + 2, 4, 4)
            worksheet.write(2 * i + 1, 4, 4)
        j = i - 1
        sum = 1
        while table.cell(2 * j + 1, 2).value != table.cell(2 * i + 1, 1).value and j >= 0:
            if (table.cell(2 * j + 1, 1).value == table.cell(2 * i + 1, 1).value):
                sum = sum + 1
            j = j - 1
        worksheet.write(2 * i + 1, 5, sum)
    book.save('DataSet2.xls')


#
def CalcB2B(team_name, date, visitor, host, winner):
    sum = 0
    win = 0  #
    sum0d = 0
    win0d = 0  #
    sum1d = 0
    win1d = 0  #
    sum2d = 0
    win2d = 0  #
    sum3d = 0
    win3d = 0  #
    for i in range(len(visitor)):
```

```
        if visitor[i] == team_name or host[i] == team_name:
            for j in range(len(visitor)):
                # if date[j]==date[i]-timedelta(days=1) and visitor[j]==team_name:
                #    sum+=1
                #    if winner[j]==team_name:
                #        win+=1el
                if date[j] == (date[i] + timedelta(days=1)):
                    if visitor[j] == team_name or host[j] == team_name:
                        sum0d += 1
                        if winner[j] == team_name:
                            win0d += 1
                elif date[j] == (date[i] + timedelta(days=2)):
                    if host[j] == team_name or visitor[j] == team_name:  # elif
                        sum1d += 1
                        if winner[j] == team_name:
                            win1d += 1
                elif date[j] == (date[i] + timedelta(days=3)):
                    if host[j] == team_name or visitor[j] == team_name:
                        sum2d += 1
                        if winner[j] == team_name:
                            win2d += 1
                elif date[j] == (date[i] + timedelta(days=100)):
                    if host[j] == team_name or visitor[j] == team_name:
                        sum3d += 1
                        if winner[j] == team_name:
                            win3d += 1
    print(team_name)
    print(win0d,sum0d)
    return win0d,sum0d

#
def CalcWin(team_name,winner,games):
    win=0;j=0
    for i in range(len(games)):
        if name_list[i] == team_name:
            j = i
    for i in range(len(winner)):
        if winner[i] == team_name:
            win = win + 1
    return win / games[j]


#
date, visitor, host, winner = read_excel(input_file_name)
date = DateRegular(date)
games,gameBefore,gameAfter = getgames(name_list, visitor, host,date)

#SheetInit(date, visitor, host, winner)
for i in range(len(name_list)):
    CalcB2B(name_list[i], date, visitor, host, winner)
    print(CalcWin(name_list[i],winner, games))
```

Appendix B Python code for Neural Network Model

This part of the code implements the establishment of a single hidden layer feedforward neural network

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
W1 = np.random.randn(1,0)
print(W1.shape[0])


# In[2]:


# -*- coding: utf-8 -*-
#wxwx
import numpy as np

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random input and output data
x = np.random.randn(N, D_in)
y = np.random.randn(N, D_out)

# Randomly initialize weights
w1 = np.random.randn(D_in, H)
w2 = np.random.randn(H, D_out)

learning_rate = 1e-6
for t in range(500):
    # Forward pass: compute predicted y
    h = x.dot(w1)
    h_relu = np.maximum(h, 0)
    y_pred = h_relu.dot(w2)

    # Compute and print loss
    loss = np.square(y_pred - y).sum()
    print(t, loss)

    # Backprop to compute gradients of w1 and w2 with respect to loss
    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.T.dot(grad_y_pred)
    grad_h_relu = grad_y_pred.dot(w2.T)
    grad_h = grad_h_relu.copy()
    grad_h[h < 0] = 0
    grad_w1 = x.T.dot(grad_h)
```

```
    # Update weights
    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2


# In[3]:


import numpy as np
import xlrd
import xlwt
import matplotlib.pyplot as plt
def sigmoid(n):
    return 1.0/(1+np.exp(-n))

workbook=xlrd.open_workbook("d:\DataSet.xls")
table = workbook.sheet_by_name('sheet')
#
X_train=np.zeros(shape=(2000,4))
Y_train=np.zeros(shape=(1,2000))
for i in range(1,2001):
    for j in range(4,8):
        X_train[i-1][j-4]=float(table.cell(i, j).value)#4*2000200040,1,2,30
    Y_train[0,i-1]=int(table.cell(i,8).value) #1*2000200001
X_train=np.transpose(X_train)
#np.random.seed(2) #
W1 = np.random.randn(4,4) * 0.01#(4*4)w1*x0(4*2000)
b1 = np.zeros(shape=(4, 1))
W2 = np.random.randn(1,4) * 0.01#(1*4),w2*w1*x0(1*2000)
b2 = np.zeros(shape=(1, 1))
#print(X_train)
#print(Y_train)
learning_rate = 0.01#

for i in range(2001):
    #
    Z1 = np.dot(W1 , X_train) + b1
    A1 = np.tanh(Z1)
    Z2 = np.dot(W2 , A1) + b2
    A2 = sigmoid(Z2)
    m=2000

    #
    logprobs = np.multiply(np.log(A2), Y_train) + np.multiply((1 - Y_train), np.log(1 - A2)
    cost = - np.sum(logprobs) / m
    cost = float(np.squeeze(cost))

    #
    dZ2 = A2 - Y_train
    dW2 = (1 / m) * np.dot(dZ2, A1.T)
    db2 = (1 / m) * np.sum(dZ2, axis=1, keepdims=True)
    dZ1 = np.multiply(np.dot(W2.T, dZ2), 1 - np.power(A1, 2))
    dW1 = (1 / m) * np.dot(dZ1, X_train.T)
    db1 = (1 / m) * np.sum(dZ1, axis=1, keepdims=True)
```

```
    #
    W1 = W1 - learning_rate * dW1
    b1 = b1 - learning_rate * db1
    W2 = W2 - learning_rate * dW2
    b2 = b2 - learning_rate * db2

    if i % 1000 == 0:
            print(": %i   %f" % (i,cost))



# In[4]:


#
X_test=np.zeros(shape=(120,4))
Y_test=np.zeros(shape=(1,120))
for i in range(2001,2120):
    for j in range(4,8):
        X_test[i-2001][j-4]=float(table.cell(i, j).value)#4*2000200040,1,2,30
    Y_test[0,i-2001]=int(table.cell(i,8).value) #1*2000200001
X_test=np.transpose(X_test)

Z1 = np.dot(W1 , X_test) + b1
A1 = np.tanh(Z1)
Z2 = np.dot(W2 , A1)+ b2
A2 = sigmoid(Z2)
Y_prediction=np.zeros(shape=(1,120))
true_case = 0#
for i in range(A2.shape[1]):
        Y_prediction[0,i] = 1 if A2[0,i] > 0.5 else 0
        if Y_prediction[0,i] == Y_test[0,i]:
            true_case = true_case+1

print(float(true_case)/120)


# In[5]:


accuracy = [0.608333333333333,0.5583333333333333,0.5583333333333333,0.5583333333333333,0.5
plt.plot(accuracy)
plt.ylabel('test accuracy')
plt.xlabel('number of tests')
plt.title("test times and test accuracy")
plt.show()


# In[6]:


#
import matplotlib.pyplot as plt
```

```
kkk = []
for i in range(1,99):
    X = [[i],[0],[2.4153],[2.4153]]
    Z1 = np.dot(W1 , X) + b1
    A1 = np.tanh(Z1)
    Z2 = np.dot(W2 , A1) + b2
    A2 = sigmoid(Z2)
    #print(A2)
    kkk.append(float(A2))
#print(kkk)
plt.plot(kkk)
plt.ylabel('Win Rate')
plt.xlabel('game density (interval days)')
plt.title("game density and winning rate")
plt.show()
```

Appendix C Detailed Game Schedule

Due to the space limitation of the paper, please refer to the file "GameSchedule.xlsx" for the detailed competition schedule

Appendix D Other Annexes

The excel sheet used in the solution process has been attached to the paper. They are "winRate18-19.xlsx" for 2018-2019 win rate, "BackToBack.csv" for back to back win rate, and "allstar.xlsx" for win rate related to all-star.