|  |  |  |
| --- | --- | --- |
| **Problem Chosen** D | **2020 MCM/ICM Summary Sheet** | **Team Control Number** 2009664 |
|  |  |  |

**Summary**

We all know the famous "barrel effect", that is, the amount of water in a barrel depends on the lowest board. This brings us to the operation of a team. Whether a team can achieve good development is closely related to many factors, such as the degree of cooperation between team members, and excellent leaders. This time we received a request from the Huskies coach to build a suitable model to understand the team's dynamics, especially to explore how the complex interactions between players on the scene affect their success. Identify helps identify specific strategies that can improve teamwork next season.

In the following modeling process, in order to reduce the impact of other parameters on the evaluation results, we assume that each team is not affected by other external environments in the game, such as regardless of weather, player physical conditions, fairness of the game, etc. This assumption helps us to use the data analysis effectively to get the model.

The first model is our own innovative model, which defines the influence of the game's mentality on the game's winning or losing, and then gives the change of the error rate with the event, and then concludes that the mentality can only determine the game's winning or losing to a small extent The main thing is to look at the strength, and then we use the neural network to train the game data, get the training results, and then give the reference values ​​of several parameters to win the game. This neural network training method is universal for a given training set for predictive classification. Next, we also analyzed the macro factors and concluded that Coach II won more games when he was on the court and had a higher win rate at home.

Contents

[1 Introduction 3](#_Toc32894578)

[2 Purpose 3](#_Toc32894579)

[3 Assumptions 3](#_Toc32894580)

[4 Developing the model 4](#_Toc32894581)

[4.1 Differential equation model 4](#_Toc32894582)

[4.1.1 Initial analysis 4](#_Toc32894583)

[4.1.2 Data analysis 6](#_Toc32894584)

[4.1.3 Differential equation model 7](#_Toc32894585)

[4.2, Developing a gray model (BP neural network) 9](#_Toc32894586)

[4.2.1 Preliminary establishment 9](#_Toc32894587)

[4.2.2 Optimizing Neural Networks 10](#_Toc32894588)

[4.2.3 Process of data training 10](#_Toc32894589)

[4.2.4 the final result 11](#_Toc32894590)

[4.3 Other factors to consider 12](#_Toc32894591)

[5 Pros and cons of the model 12](#_Toc32894592)

[5.1 BP neural net work 12](#_Toc32894593)

[5.2 Differencial equation model 13](#_Toc32894594)

[6 Summary of optimization plans for the Huskies team: 13](#_Toc32894595)

[7 Promotion of the optimal team model: 14](#_Toc32894596)

[8 Appendix 14](#_Toc32894597)

[8.1 The code 14](#_Toc32894598)

[8.2 Training set of neural network 17](#_Toc32894599)

1 Introduction

When we see the magic of a magician turning one coin into two, we want to reveal the secret. Similarly, the magic of 1 + 1> 2 presented by teamwork is something we want to explore. Research on teamwork has led to the development of many fields. Whether it is a scientific research project that tackles difficulties together, or a football match that fights side by side, teamwork is of great significance. An excellent team can make the best of each member of the team, make up for the shortcomings, and maximize the team's benefits. Factors affecting teamwork include many aspects, such as the reasonable division of labor among team members, the tacit cooperation between teammates, and the balance between personal performance and team performance. In many types of teamwork, we explore the factors that maximize the benefits of a team, and give a model for evaluating the performance of a team.

**2 Purpose**

At the invitation of Coach Husky, we first established a model to quantify and structure various influencing factors to find a combat strategy tailored for the Husky team.

We want to build a network based on passing data with players as nodes, quantify the impact of various team cooperation performances on team performance in the team cooperation, and then develop a set of performance scoring mechanisms. Find a battle strategy tailored for the Husky team. We will also deeply explore the advantages of teamwork and explore how to improve the efficiency of teamwork through the football competition, a group competitive project.

**3 Assumptions**

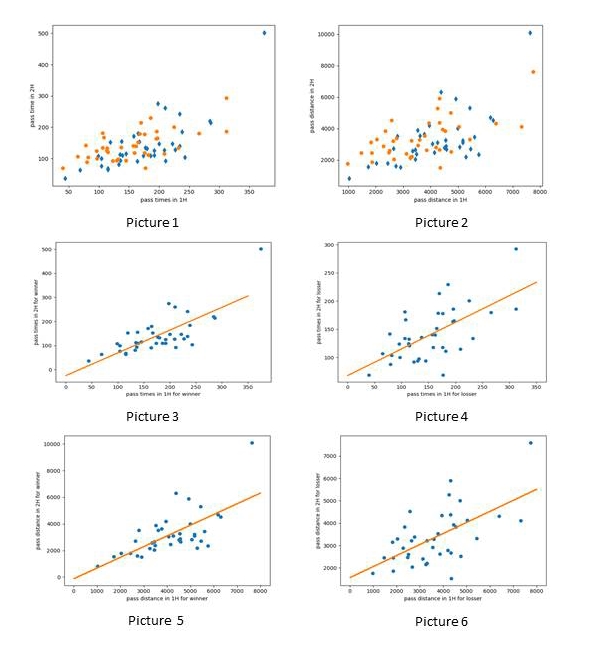
To ensure the accuracy of the model, we make the following assumptions:

* Assume that the state of each team player during each game is not affected by other external factors. For example, the weather conditions are good during each game, and it does not affect the normal performance of the game, and each player is in good physical condition and stable in each game.
* Assume that the referee is fair and fair in each game without misjudgment.
* Assume that there are no events other than the types of passes and events given in the table.

**4 Developing the model**

4.1 Differential equation model

4.1.1 Initial analysis

First, we calculate the number of passes made by the two teams in the first and second half, respectively, and the total distance that the first half and the second half of football past. According to the data of each football match, we use the number of passes in the first half of each field as the abscissa and the number of passes in the following Alf as the ordinate to draw a scatter plot (for data, please refer to Appendix（）)

The above picture includes six sub-pictures, which represent the first half and the second half of the linear fit of the winner and loser passes (the slope of the winner is 0.946, and the slope of the loser 0.4739 is significantly smaller than the winner), and the winner and loser Linear fit of the first and second half of the ball distance (winner slope 0.80, loser slope 0.496)

We can observe that they have a clear linear relationship. Then we use the least square method to fit them.

Interestingly, we found that the slope of the fitted graph of the winning team's first-half passes is 0.946, and the slope of the fitted graph of the aggregate passing distance is 0.80. However, analyzing the data of the losing team, we found that the slope of the fitting graph on the number of passes was 0.496, which is much smaller than the winning team. We can infer that the players of the losing team tend to perform worse in the second half than in the first-half. Based on this, we suspect that it may be because the players of the team that ultimately failed lost pressure in half as the ordinate to draw a scatter plot (for data, please refer to Appendix)

The first-half of the game, resulting in a negative mentality in the second half of the game and losing the game. In the next section, we will use data to prove conjectures.

### 

4.1.2 Data analysis

We define:

Error rate = number of lapsed events / total number of incidents ；

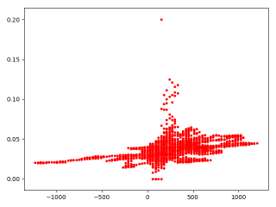
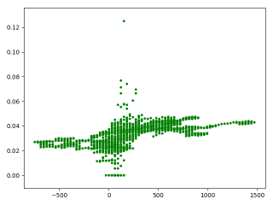
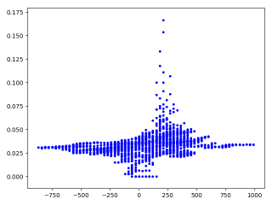
Mentality is equal to fault rate / stress.；

Stress calculation: Once a negative event occurs on our side, or a positive event occurs on the other side, our total pressure will increase by 30 (a total of 30 players, each adding a little pressure); a positive event occurs on our side, or a negative event occurs on the other side Once the incident, our pressure was reduced by thirty. If the positive correlation between stress and turnover rate is obvious, the mentality is not good. On the contrary, if the correlation coefficient is small, the mentality is good.

Define the classes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | positive category | negative category | competition | neutral events |
| Even type | Shot | Foul, Offside | Free Kick, Duel | Free Kick, Others, the ball, Goalkeeperleaving, Save attempt, Substitution, Interruption, PassFourth. |
| Event SubType | Goal kick, Shot, Free kick shot, Penalty, Smart pass  Second, the negative categories | Touch, Foul, Corner, Late, card, foul, Protest, Hand, foul, Violent, Foul, Out of game, Time, foul, Simulation | Air duel, Head pass, Ground ball, duel, Simple pass, Launch, High pass, Ground defending duel, Free kick cross, Whistle |

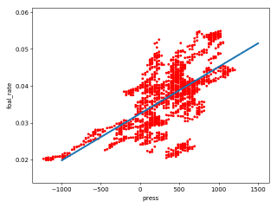
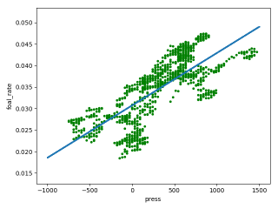
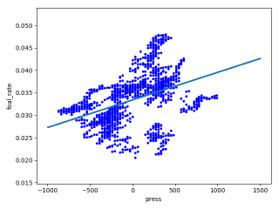
We statistics and analyze the data to get the following data chart:



The figures above shows the turnover rate

in the win, the draw and the defeat respectively.

We found no significant differences in the distribution of the three sets of graphs. Let's analyze the reasons for this set of data. The initial pressure is set to 150. If we made mistakes in the first few events, the error rate at this time is very high or even 1 (if the first event is negative event). Therefore, the data appearing in the early part of the game is not reliable and has no research value, so we exclude the first 50 data of each game and process the data again.



The figures above shows the turnover rate

in the win, the draw and the defeat respectively.

After processing in this way, our data analysis has improved considerably, and we can see that the error rate is positively related to stress. With each additional negative event, stress increases by 30. Converting it, multiplying the slope by thirty times, we can get the increase in the probability of errors for each negative event. In this way, comparing the slopes of these pressure-turnover ratios, we can find that the slope of the win is an order of magnitude smaller than that of the loss or the draw. In other words, the winners tend to have a more stable mindset. They will not be proud of victory and discouraged by defeat.

Next, we will establish a differential equation model, explore the relationship between the number of errors and the total time of events, and then use the three slopes in the pressure-error rate graph to analyze the impact of mentality.

4.1.3 Differential equation model

n order to better explain the relationship between error rate and pressure, we constructed a differential equation model

First, we assume the following parameters are continuous and smooth .At the same time, we ignore ordinary events, assuming only positive events and negative events, the consequence of this is a greatly amplified error rate (because the denominator is greatly reduced)

Then we give expressions for the following important parameters:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Foul rate | Foul time | Press | Positive events time | Times of total events | Mentality |
| Symbol | FR | FT | P | PT | T | k |

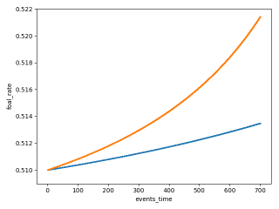
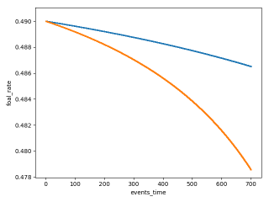
We know that: T = FT + PT, , , P=P0+30FT-30PT

Because , ，we can get ：

With , we can get:

Simplified：

It is known through analysis that the magnitude of k is very small, between 10 ^ -5 and 10 ^ -6, so it can always be guaranteed that (1-k \* T) is approximately 1 during the game time.dFR / dT = 30k \* (2FR – 1), when the turnover rate exceeds 0.5, the turnover rate increases with the increase in the number of games. At this time, the team with a better mentality will have less influence and will be able to maintain the victory under the advantage. If the mentality is not good, it may be overturned by opponents. We use the fourth-order Runge Kutta method to draw the curve (below):

The left is the image with error rate above 0.5,the right is the mage with error rate below 0.5

This can further explain that when the mentality is better, the turnover rate of fluctuations is not large, and the team can better play the real level.

4.2, **Developing a gray model (BP neural network)**

We build artificial neural network models and perform innovative optimization based on their originals. We use an artificial network model to study the influence of each influencing factor on the winning or losing of a game. The artificial neural network model has the characteristics of distributed storage and parallel collaborative processing of information. Through this model, we can get a gray system model that can judge the winning rate.

we define the important parameter here:

**win rate** = (Number of goals scored by our team-Number of goals scored by the opponent) / (Number of goals scored by our team + Number of goals scored by the opponent)

4.2.1 Preliminary establishment:

When considering the complex interaction on the football field, we first consider the performance that directly affects the winning or losing of the game. We have separately calculated the labels of the teams that have won and the teams that have lost: Total number of passes in the first half, total number of passes in the second half, total distance in the first half, total distance in the second half, total distance in the second half, and the number of different types of passes. Using this as an input, the pass rate is the training result to form a training set, and then the data is normalized and put into a neural network for training to get a trained model.

Direction of inquiry: Next, we further explore the implicit influence factors. First, we give hypothetical factors and give a reasonable explanation for the underlying reasons behind them. There are the following points: 'Total number of first half passes', 'Total number of second half passes', 'Total distance of first half passes' , 'Total distance of second half passes', 'Number of different types of passes'. Then we calculated the tag data of each winner and each tag data of the losers from the original data.

4.2.2 Optimizing Neural Networks

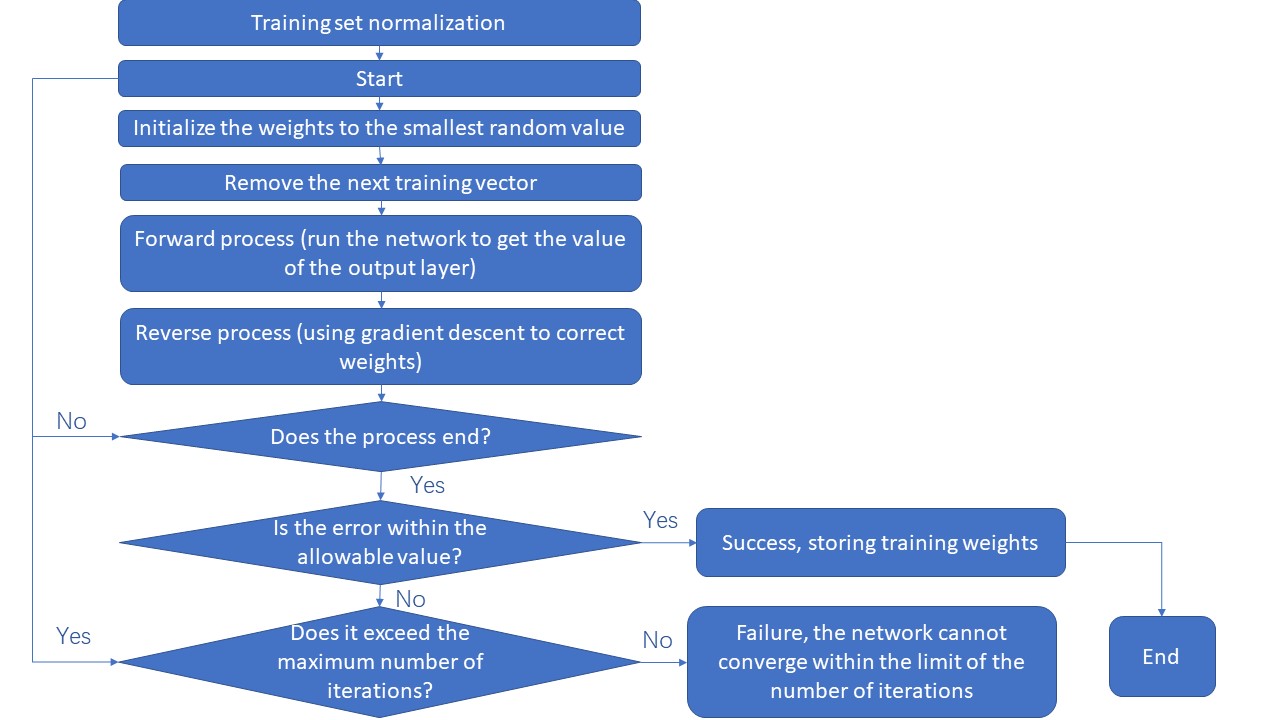
We have adopted a method of normalizing data. The original model will have problems processing this set of data , resulting in training failure to converge. So we change our mind and normalize the data. The method of data normalization is:

We denote the mean square error as Delta and the date as the data value.

Let Date = (x-Data\_mean) / derta

The advantage of this processing is that it can convert the data of a certain feature into a training set with a mean value of 0 and a variance of 1, thereby speeding up the convergence speed.

After optimization, the flow chart of the model is as follows:



4.2.3 Process of data training

Next, we trained the data. We will prove that our model is convergent by detecting the correction process of one of the weights and the change of the error in the neural network. The left figure 1 shows the weight change from the first neuron in the output layer to the first neuron in the hidden layer with the weight as the ordinate and the number of training times as the abscissa. From the graph observation, the final weight is converged to 2.5. The right figure 2 uses the number of trainings as the abscissa and the error value as the ordinate, showing the process of the error gradually converging to zero after learning 25,000 times

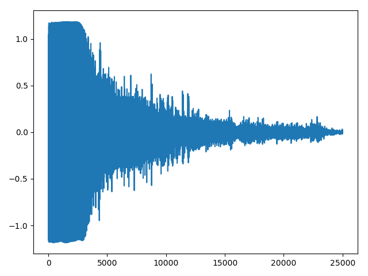
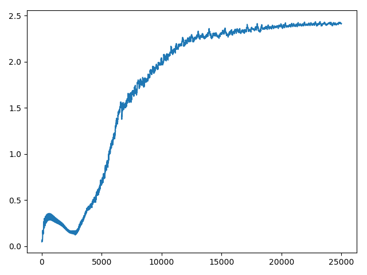


figure1and figure 2

The optimal data obtained by this neural network after learning 25,000 times is: [129.0776809068127, 77.9958890234888, 5314.844617894659, 1868.8745314494613, 13.726945784017213, 186.53819822598442, 0.9711863561572205, 10.32631199340484, 1.14508868180727, 2.4371190073318174, 1.7708923457618795]

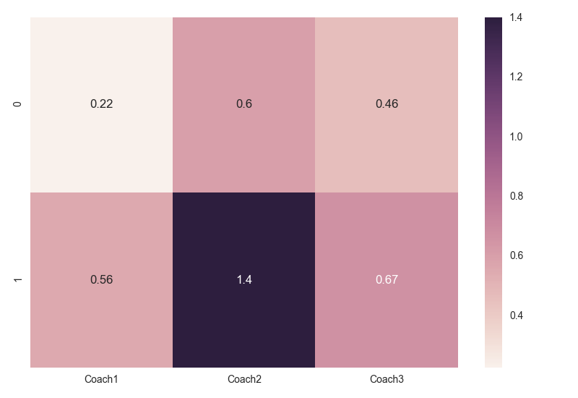
That corresponds:[129,78,5315,1869,13,187,1,10,1,2,2].129 passes in the first half, 78 passes in the second half, 5314.8 passes in the first half, 1868.8 passes in the second half, head pass 13, simple pass 186.5, launch pass 1, high pass 10, hand pass 1, smart pass 2 cross Pass 2 times can make the winning rate 0.95916474151444023 Collect huskie data and compare with model results ,Huskies data is rounded after averaging[150, 125, 3609, 2966, 15, 230, 6, 15, 3, 2, 3]

4.2.4 the final result

After learning 25,000 times in the neural network, we get the original optimal data.(in Appendix 8.2)

Then perform the contrary normalization on the optimal results output by the network to obtain the actual data. After approximate rounding, we obtain the optimal team cooperation data set that is most conducive to the team's good performance.

**4.3** Other factors to consider



0 means home, 1 means away

(a)coaching factors: We found that in both the home and away situations, the average score ranking (high to low) of the team led by the three coaches was: coach two> coach three> coach one. In particular, the average scores under the leadership of coach two were significantly higher than those under the leadership of the other two coaches. This shows that Coach 2 teaches better or that he knows the Huskies better, and initially shows that under the guidance of Coach Two, the Huskies are more likely to win the game.

(b)Home and Away factors: By comparing the average scores of home and away in each coach-led game, we know that under the leadership of different coaches, the average score of our team at home is greater than the average score of away. , Indicating that our team has an advantage in home games.

**5 Pros and cons of the model**

5.1 BP neural net work

Advantage: The training results of the neural network model we use have high accuracy, can accurately predict the gains and losses in the game, and give a certain overall improvement direction.

Disadvantage: It is difficult to determine the number of hidden layers in the neural network. The number of hidden layers is too small to predict accurately. An increase in the number of hidden layers may overfit and make predictions meaningless. By manually adjusting the number of hidden layers and observing the change of the error and its convergence speed, it is finally determined that this layer is about 14 layers, which has a good prediction significance.

5.2 Differencial equation model

Advantages: We quantify the impact of mentality on the game in an innovative way, and analyze the internal factors that affect the game.

Disadvantage of Mode 2: The differential equation model of the mindset is based on the assumption that the error rate is related to the continuity of the event, but is actually discrete. Differential equations are only instructive and have no ability to calculate accurately. And the mentality we define is small, which means that the mentality can only affect the game to a small extent. The impact is not even as great as random events. Adjusting the mindset can only reduce the error rate within a small range. It depends more on the strength of the team.

**6** Summary of optimization plans for the Huskies team**:**

Through differential equation model analysis, we found that when the husky team members' mentality is better, the fault rate does not fluctuate much, and the team can better play the real level. So we should find ways to keep each player in a good state of mind.

Through the neural network model, we get a total of 129 passes in the first half, a total of 78 passes in the second half, a passing distance of 5314.8 meters in the first half, a passing distance of 1868.8 meters in the second half, and a head pass of 13 times, simple Pass: 186.5 times, launch pass: 1 time, high pass: 10 times, hand pass: 1 time, smart pass: 2 times, cross pass: 2 times can make the winning rate more than 99%. Therefore, if the numerical standards of the above indicators can be reached when formulating the combat strategy, the win rate of the game can be greatly improved.

In the analysis of other factors, we found that the Husky team is more likely to win the game under the leadership of coach 2. It can be seen that the team cooperates better with coach 2, or coach 2 teaches well. Therefore, when choosing a coach, in order to make the team a better chance of winning, the team should choose coach 2.

The recommended tactics are as follows: first, the team should maintain a long-term cooperative relationship with coach two; second, the coach must find a way to stabilize the players' mentality before the game and even during the game, so that the players can reduce the fault rate. In the game, players should pass multiple passes in the first half, especially to increase the number of long passes when the ball is guaranteed. Expand offensive advantages and defensive efforts, minimize fouls, and reduce the number of hand passes. Strive for stability in the second half, focusing on defense, less long passes, more short distance passes, and using dribbling skills to make the ball as much as possible in his own team.

**7 Promotion of the optimal team model:**

By analyzing the performance of the football team, we can get a set of benchmarks to judge the team's performance and predict a set of decision-making programs that are beneficial to the development of the team, so as to better lead the team to success. In other projects that require teamwork, we can extract some indicators that have an impact on the development of the team, form a training set after normalization, and put them into the neural network for training, so that the weights are constantly self-adjusting, the errors are constantly decreasing, and finally converging An ideal model. Make accurate classification predictions to get the team's performance and lead the team's next direction so that the team can succeed.

**8 Appendix**

8.1 Training set of neural network before normalization

learning\_set\_rate = [[[284, 220, 6289.42692214808, 4524.09444798957, 12, 463, 2, 14, 3, 7, 3], 0.6],

[[65, 107, 1471.9460690018743, 2452.1965479904384, 12, 147, 1, 4, 4, 1, 3], -1.0],

[[192, 126, 4550.401485633126, 3263.3278539713288, 14, 267, 1, 22, 1, 7, 6], 0.3333333333333333],

[[104, 76, 2430.9166831688362, 1782.631711499596, 10, 150, 4, 14, 1, 0, 1], 0.0],

[[312, 186, 7320.2380683947895, 4115.062438805483, 13, 461, 1, 13, 6, 2, 2], 0],

[[132, 98, 3122.5790887913763, 2405.412305066918, 16, 184, 3, 13, 4, 2, 8], 0],

[[115, 121, 2509.570270588189, 2600.0086253723657, 11, 185, 2, 28, 4, 5, 1], -1.0],

[[168, 179, 3941.8647454509323, 4343.5556107341345, 14, 310, 2, 14, 2, 2, 3], -1.0],

[[266, 180, 6388.972541156617, 4318.480766392187, 23, 387, 10, 15, 3, 5, 3], 0],

[[174, 109, 4513.021566875792, 2784.2880139598565, 11, 241, 2, 22, 1, 2, 4], 0.5],

[[96, 124, 2285.113909924069, 2887.929187926763, 24, 168, 11, 9, 0, 1, 7], -0.3333333333333333],

[[195, 164, 4538.664368080263, 3828.0710307341005, 13, 313, 5, 18, 1, 2, 7], -1.0],

[[182, 111, 4324.104642283454, 2661.2459520388825, 17, 258, 1, 8, 4, 1, 4], -0.0],

[[209, 115, 4753.8814212369525, 2522.5313746149436, 4, 294, 6, 13, 7, 0, 0], -1.0],

[[196, 186, 4317.224377699875, 4380.2161367383305, 16, 342, 8, 8, 4, 2, 2], -1.0],

[[133, 81, 3448.696784061912, 2030.473379353508, 16, 168, 7, 16, 1, 1, 5], 1.0],

[[136, 94, 3261.4859213216605, 2155.0632052387446, 13, 179, 2, 24, 10, 1, 1], 0],

[[97, 100, 2477.528900227274, 2471.2273276109527, 14, 149, 5, 18, 3, 3, 5], -1.0],

[[159, 118, 3857.156226144699, 2611.4603658884935, 12, 231, 12, 13, 3, 1, 5], -0.0],

[[139, 110, 3485.033511521388, 2374.489944560798, 27, 179, 7, 22, 8, 2, 4], 0.3333333333333333],

[[137, 136, 3302.958360198853, 3223.120573395727, 11, 242, 0, 11, 5, 0, 4], -0.3333333333333333],

[[78, 142, 1825.8070615365523, 3142.76913372581, 14, 182, 2, 10, 1, 6, 5], -0.5],

[[165, 152, 3785.6058076832755, 3536.307854168737, 22, 255, 14, 13, 5, 4, 4], -0.2],

[[186, 109, 4563.1658154723555, 2887.085558467931, 11, 261, 1, 14, 6, 1, 1], 0.0],

[[164, 91, 4140.903364057851, 2465.9793438459405, 20, 202, 6, 13, 5, 4, 5], 0.3333333333333333],

[[162, 140, 3546.4971044233616, 2923.990578854937, 13, 275, 3, 10, 1, 0, 0], -0.5],

[[80, 88, 1865.4184364674381, 1861.0302977485378, 11, 139, 2, 8, 3, 1, 4], -0.42857142857142855],

[[119, 152, 2786.033447077865, 3521.4797783414665, 20, 215, 12, 12, 4, 6, 2], 1.0],

[[243, 104, 5756.985209069811, 2351.7144171182154, 14, 301, 1, 19, 9, 1, 2], 1.0],

[[82, 104, 1858.0764383470428, 2446.4914081957145, 16, 140, 15, 10, 2, 1, 2], -0.0],

[[104, 100, 2027.0879429770077, 1794.3901733117475, 14, 177, 2, 1, 5, 0, 5], 0.3333333333333333],

[[238, 185, 4991.762905940994, 4004.5112708014685, 12, 391, 2, 7, 4, 5, 2], 0.6666666666666666],

[[375, 502, 7625.942132193622, 10097.310680269864, 19, 825, 0, 17, 3, 5, 8], 0.5],

[[200, 147, 5175.703801826598, 3227.197011847874, 11, 297, 8, 24, 1, 1, 5], 1.0],

[[209, 127, 5175.7880606766985, 3098.6661892562897, 15, 274, 11, 26, 4, 2, 4], 0.0],

[[108, 167, 2355.224013072094, 3837.8267365243455, 19, 229, 3, 16, 2, 5, 1], -0.5],

[[145, 94, 3257.0240315493083, 2155.3674842924493, 11, 205, 2, 10, 5, 1, 5], -1.0],

[[176, 178, 4728.283765154024, 5001.235031318259, 17, 296, 6, 16, 6, 3, 10], -1.0],

[[115, 64, 2904.935297334927, 1526.9423504405368, 16, 140, 5, 14, 1, 0, 3], 1.0],

[[135, 113, 3349.0722248159454, 2584.3983253722895, 18, 192, 17, 12, 4, 0, 5], 0],

[[130, 94, 3309.7854180584595, 2207.3470342935902, 12, 200, 1, 9, 2, 0, 0], -1.0],

[[312, 293, 7744.1838596432935, 7599.65729575475, 25, 535, 3, 25, 5, 4, 8], -0.0],

[[177, 69, 4353.736158707954, 1516.6536788779117, 11, 210, 5, 15, 1, 3, 1], -0.0],

[[115, 68, 2731.2398248812897, 1608.1822962726662, 11, 145, 9, 12, 1, 3, 2], 0.2],

[[197, 165, 4443.147108874859, 3935.8976642682555, 19, 305, 13, 17, 2, 1, 5], -1.0],

[[222, 147, 5597.194401219097, 3447.080364039, 13, 329, 1, 15, 4, 2, 5], 1.0],

[[170, 214, 4246.406723915734, 5273.3149904354705, 28, 326, 5, 10, 3, 0, 12], -1.0],

[[232, 134, 5435.395418056287, 3309.7222430111688, 19, 300, 17, 19, 5, 3, 3], -0.3333333333333333],

[[99, 108, 2651.9692334448237, 2714.6924948581063, 11, 177, 4, 12, 0, 0, 3], 0],

[[167, 153, 3614.562040900074, 3532.2985695083216, 23, 272, 4, 14, 4, 1, 2], 0.0],

[[192, 110, 4583.709131550148, 2669.211460655603, 25, 232, 10, 24, 6, 4, 1], 1.0],

[[44, 36, 1030.6865043805801, 816.3318089862648, 6, 62, 4, 5, 2, 1, 0], 0.0],

[[186, 230, 4320.216562684874, 5911.448353697324, 14, 374, 3, 9, 4, 5, 7], -0.0],

[[138, 155, 3767.3357496750814, 3624.6262902350445, 14, 255, 3, 14, 4, 0, 3], 0.0],

[[286, 215, 6179.981661406202, 4701.622398992552, 27, 437, 7, 17, 4, 2, 7], 0.42857142857142855],

[[114, 133, 2767.050772247689, 3377.5575788365995, 18, 199, 11, 15, 3, 0, 1], -0.6],

[[225, 201, 5031.651261487469, 4122.425522875113, 16, 377, 7, 14, 1, 8, 3], -0.0],

[[177, 135, 4069.2767090371235, 3035.5656180775663, 18, 275, 3, 14, 1, 1, 0], 0.5],

[[210, 261, 4917.31604127615, 5887.503230042927, 12, 430, 5, 11, 5, 5, 3], 1.0],

[[69, 63, 1724.8835206341985, 1554.3007212746597, 6, 114, 2, 6, 0, 1, 3], 0.0],

[[227, 128, 5063.787982497974, 2831.8040422375407, 11, 319, 3, 12, 3, 2, 5], 0.5],

[[157, 141, 3592.660325609934, 3289.3581319081995, 12, 267, 2, 12, 5, 0, 0], -1.0],

[[106, 181, 2574.1000893367336, 4520.163105868153, 9, 241, 5, 21, 3, 3, 5], -0.6666666666666666],

[[180, 132, 4254.9475673593815, 3103.79773964357, 20, 265, 6, 14, 0, 3, 4], 1.0],

[[198, 275, 4373.577357950337, 6313.166948986033, 19, 418, 4, 14, 4, 8, 6], 1.0],

[[40, 69, 979.4507967852833, 1754.0120692209036, 4, 87, 2, 13, 1, 2, 0], -0.5],

[[106, 134, 2040.8095880115372, 3301.694049050632, 22, 182, 4, 21, 4, 5, 2], -1.0],

[[211, 93, 5293.212720853517, 2187.5540577554707, 18, 267, 6, 9, 2, 0, 2], 1.0],

[[234, 139, 5455.550882718332, 2711.6990620699216, 14, 338, 1, 12, 1, 4, 3], 1.0],

[[123, 92, 2666.168423756904, 2045.7225384462001, 15, 180, 8, 10, 0, 2, 0], -0.3333333333333333],

[[113, 125, 2639.3077038907095, 3216.354748959563, 22, 180, 5, 22, 2, 1, 6], -1.0],

[[234, 242, 5429.493836745718, 5302.858230355031, 13, 438, 1, 11, 2, 7, 4], 1.0],

[[158, 171, 3526.617913602203, 3878.012630127751, 17, 282, 4, 10, 5, 5, 6], 0.3333333333333333],

[[176, 118, 4193.702840706255, 2791.5907830366505, 17, 246, 1, 19, 4, 4, 3], -0.3333333333333333],

[[165, 180, 3954.071864635465, 4200.464431671431, 22, 281, 1, 19, 7, 5, 10], 1.0],

[[145, 115, 3448.74468481393, 2656.767863134473, 14, 228, 1, 9, 1, 2, 5], 1.0]]

8.2 The code

# coding with python

# file name :RungleKutta

import math

from pylab import \*

import random

random.seed()

def aim\_func(\*y):

return y[1]

def Runge\_Kutta( function, intervall, step, \*args):

'''

:param function: equation which satisfied y'=f(y, x),and draw piction

:param intervall: input the interval to fraw

:param step: precision like 0.1，0.01 ...

:param args: give the initial number (x\_0, y\_0)

:return: return a list of function

'''

lenth = len(args)

# define the step

h = step

x\_n = [intervall[0] + h \* i for i in range(math.ceil((intervall[-1] - intervall[0]) / step))]

y\_n = []

y\_n.append(args[1])

frequence\_controler = 0

for i in range(len(x\_n)):

k\_1 = function(x\_n[i], y\_n[i])

k\_2 = function(x\_n[i] + (h / 2), y\_n[i] + (h \* k\_1 / 2))

k\_3 = function(x\_n[i] + (h / 2), y\_n[i] + (h \* k\_2 / 2))

k\_4 = function(x\_n[i] + (h / 2), y\_n[i] + (h \* k\_3))

q = (y\_n[i] + h \* (k\_1 + 2 \* k\_2 + 2 \* k\_3 + k\_4) / 6)

if q <0:

q = 0

elif q>1:

q = 1

y\_n.append(q)

del y\_n[-1]

scatter(x\_n, y\_n, s=0.1, marker='o')

xlabel('events\_time')

ylabel('foal\_rate')

return y\_n

# when k = 6.15146464e-06

k1 = 6.15146464 \* (10 \*\* (-6))

def fun(\*args):

return 30 \* k1 \* (2 \* args[1] - 1) / (1 - 60 \* k1 \* args[0])

# when k = 1.27035477e-05

k2 = 1.27035477 \* (10 \*\* (-5))

def fun1(\*args):

return 30 \* k2 \* (2 \* args[1] - 1) / (1 - 60 \* k2 \* args[0])

Runge\_Kutta( fun, [0, 700], 0.001, 0.00, 0.49)

Runge\_Kutta( fun1, [0, 700], 0.001, 0.00, 0.49)

show()

# for i in range(10):

# Runge\_Kutta('b', fun, [0, 300], 0.005, 0.00, 0.01 + 0.1\*i)

# Runge\_Kutta('r', fun1, [0, 300], 0.005, 0.00, 0.01 + 0.1\*i)

# show()

# file name: least\_square

import numpy as np

from pylab import \*

from full\_events\_analysing import X\_win, Y\_win, X\_tie, Y\_tie, X\_loss, Y\_loss

def least\_square(X, Y, color):

scatter(X, Y, c=color, marker='.')

R\_trans = np.array([[1] \* len(X), [i\_1 for i\_1 in X]])

R = np.transpose(R\_trans)

A = np.matmul(np.linalg.inv(np.matmul(R\_trans, R)), np.matmul(R\_trans, Y))

Y\_1 = []

for i in np.arange(-1000, 1500, 0.03):

q\_0 = np.transpose(np.array([1, i]))

q\_1 = np.matmul(A, q\_0)

Y\_1.append(q\_1)

Xn1 = np.arange(-1000, 1500, 0.03)

scatter(Xn1, Y\_1, s=0.1)

xlabel('press')

ylabel('foal\_rate')

show()

return A

print(least\_square(X\_win, Y\_win, 'blue'))

print(least\_square(X\_tie, Y\_tie, 'green'))

print(least\_square(X\_loss, Y\_loss, 'red'))

# filename:passing\_events\_analysing2

# core:nervenet

# from passing\_events\_analysing1 ,we can get our training set winner\_learning\_data + losser\_learning\_data

from passing\_events\_analysing1 import winner\_learning\_data\_rate, losser\_learning\_data\_rate, \

learning\_set\_rate\_after\_normalization, mean\_list, derta\_list,winner\_learning\_data, losser\_learning\_data

import pandas as pd

import numpy as np

import random

import matplotlib.pyplot as plt

random.seed()

matches\_data = pd.read\_csv(open('matches.csv'))

# now we define a new teacher signal's weigth

teacher\_signal\_weight = []

print(matches\_data[:3])

for i in range(38):

q = matches\_data[i:i + 1]

if q['OwnScore'][i] + q['OpponentScore'][i] != 0:

q\_num = abs(q['OwnScore'][i] - q['OpponentScore'][i]) / (q['OwnScore'][i] + q['OpponentScore'][i])

else:

q\_num = 0

teacher\_signal\_weight.append(q\_num)

print(teacher\_signal\_weight)

for i in range(38):

winner\_learning\_data[i][1] = winner\_learning\_data[i][1] \* teacher\_signal\_weight[i]

losser\_learning\_data[i][1] = losser\_learning\_data[i][1] \* teacher\_signal\_weight[i]

# define a functioin to plot each element's distribution

def ele\_point\_plot(list\_for\_learning, num):

x\_set = []

y\_set = []

for ele in list\_for\_learning:

x\_set.append(ele[0][num])

y\_set.append(ele[1])

plt.plot(x\_set, y\_set, 'd')

plt.xlabel(num, fontsize=14)

plt.ylabel('win or loss', fontsize=14)

for i in range(11):

ele\_point\_plot(winner\_learning\_data, i)

ele\_point\_plot(losser\_learning\_data, i)

plt.show()

# --------------------------------------nerve\_net\_work------------------------------------------------------------------

# nerve net of four floor

import numpy as np

import random

import matplotlib.pyplot as plt

random.seed()

# nerve net of four floor

class NerveNet():

def \_\_init\_\_(self, form\_list):

self.input\_matrix = []

self.output\_matrix = []

self.weight\_matrix = []

self.threshold\_matrix = []

self.error\_matrix = []

for i in range(len(form\_list)):

# the input floor

self.input\_matrix.append([0 for j in range(form\_list[i])])

self.output\_matrix.append([0 for j in range(form\_list[i])])

self.error\_matrix.append([0 for j in range(form\_list[i])])

if i == 0:

self.threshold\_matrix.append([0 for j in range(form\_list[0])])

else:

self.threshold\_matrix.append([random.uniform(0, 0.1) for j in range(form\_list[i])])

if i < len(form\_list) - 1:

self.weight\_matrix.append(

np.reshape(np.array([random.uniform(0, 0.1) for j in range(form\_list[i] \* form\_list[i + 1])]),

[form\_list[i], form\_list[i + 1]]))

@staticmethod

def sigmoid(x):

# define sigmoid function

return (2 / (1 + np.exp(-x))) - 1

@staticmethod

def sigmoid\_deriv(x):

return 0.5 \* (1 + NerveNet.sigmoid(x)) \* (1 - NerveNet.sigmoid(x))

@staticmethod

def list\_sigmoid(lst):

q = []

for ele in lst:

q.append(NerveNet.sigmoid(ele))

return q

@staticmethod

def list\_sigmoid\_deriv(lst):

q = []

for ele in lst:

q.append(NerveNet.sigmoid\_deriv(ele))

return q

def data\_input(self, lst):

for i in range(4):

if i == 0:

# calculate the input of input\_floor

self.input\_matrix[0] = np.array(lst)

# calculate the output of input\_floor (have not any change)

self.output\_matrix[0] = self.input\_matrix[0].tolist()

else:

# calculate the input of the first hide floor

q = np.dot(np.array(self.output\_matrix[i - 1]), np.array(self.weight\_matrix[i - 1]))

self.input\_matrix[i] = q

# calculate the output of the first hide floor

q1 = NerveNet.list\_sigmoid(np.array(self.input\_matrix[i]) - np.array(self.threshold\_matrix[i]))

self.output\_matrix[i] = q1

return self.output\_matrix[3]

def learning\_once(self, lst, speed=1):

# learning speed

alpha = 0.3 \* speed

beta = 0.4 \* speed

q\_output = self.data\_input(lst[0])

# calculate the error of floor

for i in [3, 2, 1, 0]:

if i == 3:

q = NerveNet.sigmoid\_deriv(self.input\_matrix[i][0] - self.threshold\_matrix[i][0])

self.error\_matrix[i] = [-2 \* (lst[1] - q\_output[0]) \* NerveNet.sigmoid\_deriv(self.input\_matrix[i][0] -

self.threshold\_matrix[i][

0])]

else:

self.error\_matrix[i] = np.dot(np.array(self.error\_matrix[i + 1]), self.weight\_matrix[i].T) \* np.array(

NerveNet.list\_sigmoid\_deriv(np.array(self.input\_matrix[i]) - np.array(self.threshold\_matrix[i])))

# change the weight and the threshold

for i in [2, 1, 0]:

for j in range(len(self.weight\_matrix[i])):

for k in range(len(self.weight\_matrix[i][0])):

self.weight\_matrix[i][j][k] -= alpha \* self.error\_matrix[i + 1][k] \* self.output\_matrix[i][j]

self.threshold\_matrix[i + 1] = (np.array(self.threshold\_matrix[i + 1]) + beta \* np.array(

self.error\_matrix[i + 1])).tolist()

def learning\_n\_times(self, learning\_set, n):

monitor = []

time = 0

while time < n:

for ele in learning\_set:

time += 1

self.learning\_once(ele)

monitor.append(self.error\_matrix[3][0])

return monitor

win\_net = NerveNet([11, 14, 14, 1])

Y = win\_net.learning\_n\_times(learning\_set\_rate\_after\_normalization, 27000)

X = [i for i in range(len(Y))]

plt.plot(X, Y)

plt.show()

for i in range(38):

print(learning\_set\_rate\_after\_normalization[i][1], ',',

win\_net.data\_input(learning\_set\_rate\_after\_normalization[i][0])[0])

def my\_model(lst):

return win\_net.data\_input(lst)[0]

# get the list of the optimal data which is after normalization

factor\_to\_win = []

for i in range(11):

set = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

set[i] = 1

factor\_to\_win.append(my\_model(set))

print(factor\_to\_win)

# result

factor\_to\_win = [-0.5801078679636702, -0.98495943748985304, 0.97881415897561874, -0.99880078824552943,

-0.33291779264271792, -0.68276448946141344, -0.97073702488308466, -0.69908097906941058,

-0.98332037234977021, -0.029094574401479223, -0.75407110071245342]

print('\n', win\_net.data\_input(

[-0.5801078679636702, -0.98495943748985304, 0.97881415897561874, -0.99880078824552943, -0.33291779264271792,

-0.68276448946141344, -0.97073702488308466, -0.69908097906941058, -0.98332037234977021, -0.029094574401479223,

-0.75407110071245342]

))

# anti\_normalization

final\_win\_factor\_list = []

for i in range(11):

q = factor\_to\_win[i] \* derta\_list[i] + mean\_list[i]

final\_win\_factor\_list.append(q)

print(final\_win\_factor\_list)

# result

final\_win\_factor\_list = [129.0776809068127, 77.9958890234888, 5314.844617894659, 1868.8745314494613, 13.726945784017213,

186.53819822598442, 0.9711863561572205, 10.432631199340484, 1.14508868180727,

2.4371190073318174, 1.7708923457618795]

# normalizaton

initial\_data = [129, 78, 5315, 1869, 13, 187, 1, 10, 1, 2, 2]

after\_normalizaton = []

for i in range(11):

q = (initial\_data[i] - mean\_list[i]) / derta\_list[i]

after\_normalizaton.append(q)

print(after\_normalizaton)

print('Huskies\_average\_win\_rate', win\_net.data\_input(after\_normalizaton))

print(win\_net.data\_input([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]))