Denys POTAPENKO[1]

Scientific supervisor: Tetiana TERESHCHENKO[2]

DOI: https://doi.org/10.53052/9788366249868.18

ROZWÓJ JEDNOSTRONICOWEJ APLIKACJI DO OBLICZANIA OCEN SZACHISTÓW Z WYKORZYSTANIEM SIECI NEURONOWEJ

Streszczenie: Artykuł opisuje rozwój sztucznej sieci neuronowej oraz aplikacji webowej, która pozwala na obliczenie rankingu szachisty po minimalnej liczbie gier komputerowych. Stosowane są trzy parametry: całkowita wartość rozbieżności między jakością ruchów gracza a ruchami obliczonymi przez silnik, wynik końcowej pozycji oraz liczba ruchów. Wykorzystano silnik szachowy Sztokfisz, technologię JavaScript Nodejs, Express.js.

Słowa kluczowe: aplikacja jednostronicowa, sztuczna sieć neuronowa; silnik szachowy; perceptron; ocena ELO, Node.js, Express.js.

DEVELOPMENT OF A SINGLE PAGE APPLICATION FOR CALCULATING THE RATING OF CHESS PLAYERS USING A NEURAL NETWORK

Summary: The article describes the development of an artificial neural network and a web application that allows you to calculate the rating of a chess player after a minimum number of computer games. Three parameters are used: the total value of the discrepancy between the quality of the player's moves and the moves calculated by the engine, the final position score and the number of moves. Stockfish chess engine, JavaScript technology Nodejs, Express.js were used.

Keywords: single page application, artificial neural network, chess engine, perceptron, rating ELO, Node.js, Express.js.

1. Introduction

Artificial neural networks (ANNs) are used in various fields of science, including information technology. They serve as a tool for modeling, processing and

[1] Odessa State Environmental University, Computer Science, Management and Administration Faculty, Computer Science, email: denpotap2000@gmail.com
[2] Engineering Science Ph.D., Odessa State Environmental University, associate professor at the department of information technology, tereshchenko.odessa@gmail.com

classification of large and complex data. ANNs allow you to solve complex problems in cases where conventional mathematical algorithms are ineffective, such as games and learning applications [1]. Chess is not only a sport, but also an interesting and popular game. Therefore, the gaming industry is constantly improving software products and applications for playing chess [2, 3]. When creating such applications, the urgent task is to calculate the ELO rating of chess players, which characterizes the individual strength factor of the player [4].
Similar applications and programs are used to calibrate new players in the electronic system. After all, now new users are either given a numerical value of the rating, or offered to choose the experience of playing chess from a list that can spoil the fun of the first few games. Or to determine your own rating in terms of the minimum number of games played. There are similar systems in free access, but they have significant drawbacks.
The purpose of the development is to create a software product for calculating the ELO rating of chess players, which uses effective advanced algorithms and has a web interface.
There are several programs that solve similar problems and have a high rating among players, for example, elometer.net, chessmaniac.com and lichess.org. Elometer.net is built as a website service that determines the rating of a chess player in the short term [5]. The player's moves in certain chess positions are used as input data. This website offers 76 unrelated chess positions. The site has several significant drawbacks. There is no time limit, there is no function to go again. If a situation arises that requires a page reload, the rating calculation is incorrect. For example, after repeating this operation repeatedly, the user receives the message "1035, with 95% confidence interval [899… 1170]". This figure is incorrect. It had to be either much smaller or the site had to inform the user that that the input data is insufficient for evaluation. The lichess.org service is built on similar principles and has the same shortcomings [6]. Shessmaniac.com is a website service that is a modified version of elometer.net. It has a much smaller number of tasks: 10 vs. 76 in elometer.net. However, this number of tasks is not always enough to determine the objective rating of the player. Based on the results of the analysis, it was decided to provide for the creation of the application time limit - 10 minutes for each (player and computer) per game. And choose as a basis another technology that allows, starting from the starting position, each subsequent receive after 2 moves - the player and the computer, respectively. The rating is determined by no more than 51 positions processed by the player.

2. Materials and methods

The ELO rating is calculated according to the established rules. When player A plays against player B, the mathematical expectation of the number of points he must score in this game by formula (1) is calculated:

$$E_A = \frac{1}{1+10^{\frac{R_b - R_a}{400}}}, \tag{1}$$

where *EA* - mathematical expectation of the number of points that player A will score in the game with B;
RA - old rating of player A;

RB - old rating of player B;
The new rating of player A is determined by formula (2):

$$R'_A = R_A + K \times (S_A - E_A), \tag{2}$$

where *K* - coefficient, the value of which is equal to 10 for chess players with a rating above 2400, 15 - for chess players with a rating below 2400 and 25 - for beginners (the first 15 games from receiving a FIDE rating);
SA - new rating of player A.
Creating an algorithm to determine the chess player's rating after one game with a computer is a difficult task, so a neural network was used to determine the weight of important factors [7,8].
Perceptron training is performed as follows:

- Initialization of weight matrices W (random values).
- Give input X and calculate the output Y for the target vector YT.
- If the solution is correct - go to the next step; otherwise calculate the difference by formula (3):

$$D = YT - Y\ . \tag{3}$$

Modify the scales according to formula (4):

$$w_{ij}(e+1) = N_{ij}(e) + \alpha D X_i\ , \tag{4}$$

where *Wij (e)* - weight value from neuron i to neuron j before debugging;
wij (e + 1) - weight value after adjustment;
α - learning speed ratio;
Xi - input of neuron i;
e - epoch number (iteration during training).
Follow the loop from the input feed step until the network stops making mistakes.
Calculate the answer by formula (5):

$$OUT = \frac{1}{1+e^{-C \times NET}}\ , \tag{5}$$

where *C* - the coefficient of the width of the sigmoid along the abscissa;
NO- summing block.

3. Results and discussion

The project consists of a neural network, client and server parts, installed modules: Express and Stockfish [9,10]. The block diagram of the project is shown on Fig. 1. The entire client part is in one separate directory. Its contents are sent by a server that can be located remotely. There are files in .html, .css, .js format.
The graphical user interface includes (Fig. 2):

- board with figures;
- two chess timers for each player with a set time of 10 minutes;
- two toggle buttons to select a color for the game;
- button to start a new game, a button for the player to admit his defeat;
- a list that opens with the names of the figures into which you can turn a pawn;

- list of moves made in PGN format, dialog box with a message about the player's rating.

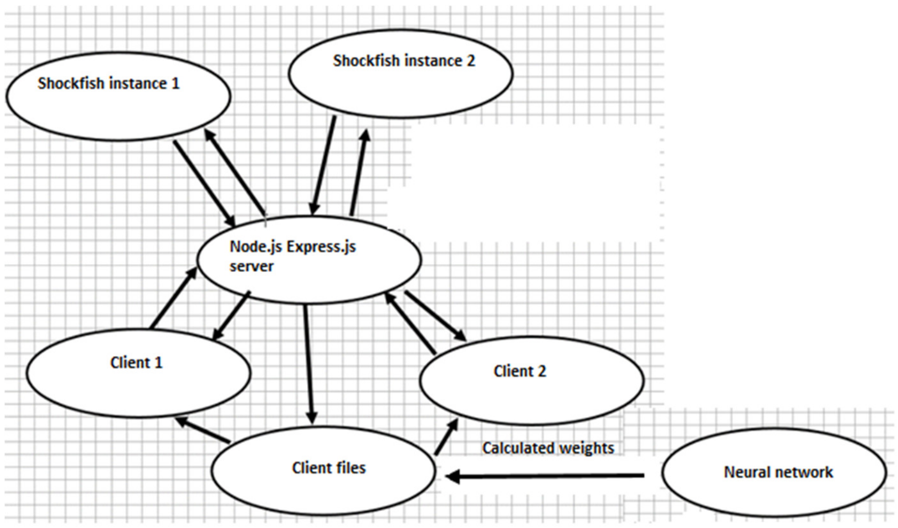
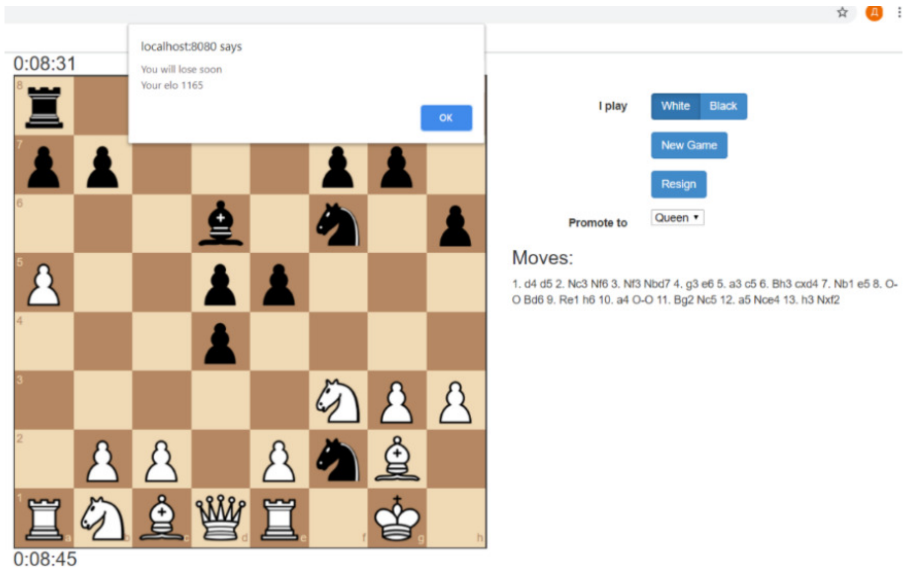


Figure 1. Block diagram of the project



localhost:8080 says

You will lose soon
Your elo 1165

Moves:

1. d4 d5 2. Nc3 Nf6 3. Nf3 Nbd7 4. g3 e6 5. a3 c5 6. Bh3 cxd4 7. Nb1 e5 8. O-O Bd6 9. Re1 h6 10. a4 O-O 11. Bg2 Nc5 12. a5 Nce4 13. h3 Nxf2

Figure 2. Graphical user interface

The button to start a new game will return the position to the starting position. The button to admit your defeat becomes available after the first move made by the player. When the html-page is loaded, a script is executed, which receives a unique identifier from the server. After each player's move or from the beginning of the player's game, the black pieces are sent a request to the server with the following parameters: unique client ID, color of the player's pieces, logical parameter to run one or two position analyzes, game history in the form of moves. After the request, the connection to the server is checked. In case of its absence the corresponding message is displayed on the screen. After this request, the computer timer is turned on and the server is expected to respond with the following parameters: computer speed; the best move for the player in the last position after the move of the computer; two chess position estimates (before and after the player's last move).

The party is terminated after the computer runs, provided:

- one of the parties ran out of time;
- position score is more than +5 in favor of the computer;
- the position score is more than +5 in favor of the player and the previous position before the player's move also had a score of +5 in his favor;
- more than 101 moves are made in the game if the player plays for whites, or more than 100 moves if a player plays for blacks;
- the player pressed the button to acknowledge his defeat.

At the end of the game, the difference between the best moves and the player's moves is calculated. In this case, the chess engine can make a big difference between the positions, even if the player's move is the best. Therefore, the odds change if the player's move does not match the best.

Next, the advantage factor is calculated. The party ends early provided:

- advantage on the side of the computer, the player admitted his defeat or the player ran out of time, with the coefficient assigned to -5;
- the advantage on the player's side is a factor of 5.

In the event that neither side has achieved an advantage of more than 5 points, the coefficient is assigned an estimate of the last position.

The last factor is the number of moves factor. When a player loses and a small number of moves are made in the game, that player plays worse than the loser, but lasts longer. It is better for both of them to play the player who wins the computer. The best player is the one who wins and makes the fewest moves. Therefore, the last coefficient is calculated by dividing the unit by the number of moves in the game and adding a minus in front, if the coefficient of advantage is less than zero. If the odds ratio is zero, then the odds ratio will also be zero. Thus, the input parameters are optimized so that all three parameters are in the range [-5; 5]. The player's ELO rating is determined by having as parameters the obtained 3 coefficients and the corresponding weights,

If a player plays for black pieces, he is added a certain number of points. The determined rating will be the closest to the rating of a chess player in a blitz party. If a player plays very poorly his rating does not count.

The neural network in this project is a multilayer perceptron with an input layer of 3 neurons, a hidden layer also of 3 neurons and an output layer of one neuron. The perceptron uses the principle of learning with a teacher with the weights of neurons, which from the beginning are assigned a random value [11].

The developed perceptron is presented on Fig. 3.

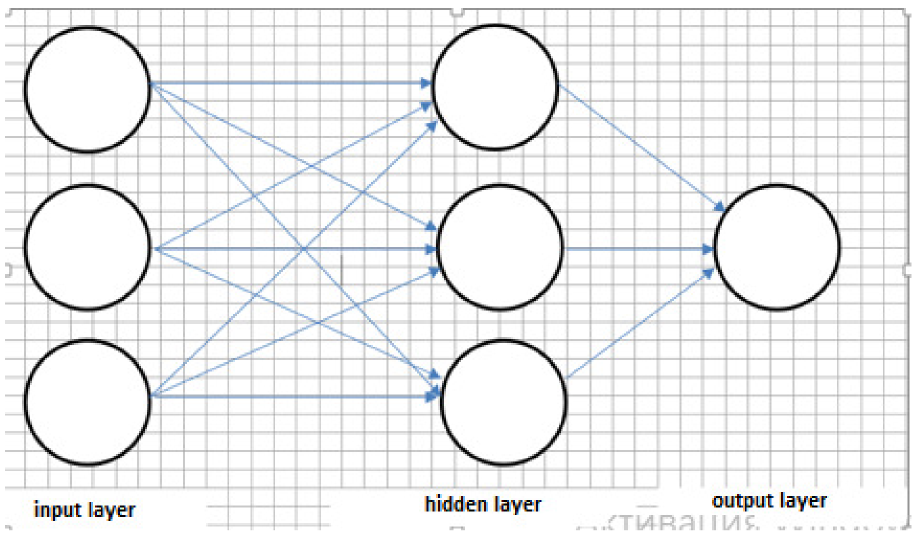


Figure 3. Perceptron

Neural network training follows the delta rule until the error increases. The sigmoid is used as a function of activation. The Android computer chess.com application is used as a player to receive the data set. Levels from 800 to 2900 ELO with a step of 150 points were tested. Less (2 lines) of data was recorded after playing with computers with an ELO of less than 1700, because they are less stable in terms of game quality. 6 lines are written for higher levels. After completing the training, you can enter coefficients for testing.
The results of network training are presented in table 1. The results of checking the correctness of the rating calculation are presented in table 2.

Table 1. Network learning results

№ learning step	Network error	The ratio of the best moves with the moves of the player	Coefficient of advantage	The ratio of the number of moves
1	0.315329	0.39683	0.196941	0.496094
2	0.314404	0.393664	0.193891	0.492197
3	0.313461	0.390502	0.19085	0.488309
4	0.3125	0.387345	0.187818	0.484431
5	0.31152	0.384193	0.184796	0.480563
10	0.306329	0.368507	0.169837	0.461377
20	0.294343	0.337563	0.140791	0.423894
30	0.279971	0.307322	0.11318	0.387864
40	0.263204	0.27797	0.0873802	0.353647
50	0.243783	0.249732	0.0637954	0.32163
60	0.223218	0.222856	0.0428067	0.292187
70	0.201396	0.197598	0.0247085	0.265623
80	0.180474	0.174192	0.00965386	0.242124
90	0.162529	0.152809	-0.00237057	0.221729
100	0.146432	0.133539	-0.0115329	0.204331

1000	0.0530457	0.000247121	0.0861072	0.044769
2000	0.0505432	0.00845809	0.111903	0.015771
3000	0.0500267	0.0102909	0.117811	0.00922604
4000	0.0499085	0.0107103	0.119176	0.00772018
5000	0.0498812	0.0108071	0.119492	0.00737206
6000	0.0498749	0.0108295	0.119565	0.00729149
7000	0.0498735	0.0108347	0.119582	0.00727284
8000	0.0498731	0.0108359	0.119586	0.00726852
9000	0.049873	0.0108361	0.119587	0.00726752
10000	0.049873	0.0108362	0.119587	0.00726729
11000	0.049873	0.0108362	0.119587	0.00726723
12000	0.049873	0.0108362	0.119587	0.00726722
13000	0.049873	0.0108362	0.119587	0.00726722
14000	0.049873	0.0108362	0.119587	0.00726722
15000	0.049873	0.0108362	0.119587	0.00726722
16000	0.049873	0.0108362	0.119587	0.00726722
17000	0.049873	0.0108362	0.119587	0.00726722
18000	0.049873	0.0108362	0.119587	0.00726722
18251	0.049873	0.0108362	0.119587	0.00726722

Table 2. The results of checking the correctness of the rating calculation

PGN	Player side	Expected	Actual
1. Nf3 d6 2. d4 Nf6 3. c4 g6 4. Nc3 Bg7 5. e4 Nc6 6. h3 OO 7. Be3 e5 8. d5 Ne7 9. Be2 Nh5 10. Qd2 f5 11. Bh6 fxe4 12. Nxe4 Nf4 13. Bxg7 Kxg7 14. OO Nf5 15. c5 Qe7 16. Bc4 Bd7 17. b4 Nh6 18. Nfg5 Rf5 19. h4 Nf7 20. Ne6 + Nxe6 21. dxe6 Bxe6 22. Bxe6 Qxe6 23. Ng3 Rf4 24. h5 g5 25. Rfe1 h6 26. Re4 d5 27. Rxf4 exf4 28. Qd4 + Qe5 29. Nf5 + Kh7 30. Qxe5 Nxe5 31. Re1 Re8 32. Nd4 c6 33. Kf1 Re7 34. Nf3 Kg7 1-0	White	2070	2100
1. e4 c5 2. Nf3 Nc6 3. d4 cxd4 4. Nxd4 e5 5. Nxc6 bxc6 6. Nc3 Nf6 7. Bg5 Be7 8. Bxf6 Bxf6 9. Bc4 OO 10. Bb3 a5 11. OO Ba6 12. Re1 Qb6 13. Qxd7 Rad8 14. Qf5 Rd2 15. Rad1 Bc8 16. Qf3 Rfd8 17. Rxd2 Rxd2 18. Rd1 Qd4 19. Rxd2 Qxd2 20. Qd1 Qg5 21. g3 Bg4 22. Qe1 h5 23. Qe3 Qg6 24. f3 Bg5 25. Qd3 Bh3 26. Kf2 Be7 27. Na4 Qf6 28. Qc3 h4 29. Qxa5 hxg3 + 30. hxg3 Qh6 31. Qe1 Bd7 32. Kg1 c5 33. Nc3 Qh3 34. Nd5 Bf8 35. Bc4 Be6 36. Qf2 Qh6 37. Qe1 g5 38. Qe3 g4 39. Qxh6 Bxh6 40. Nf6 + Kg7 41. Bxe6 Kxf6 42. Bxg4 c4 43. Kf2 Bc1 44. b3 cxb3 45. cxb3 Ba3 46. Ke2 Ke7 47. Kd3 f6 48. Kc4 Kd6 49. b4 Bb2 50. a4 1-1	White	1926	1900
1. d4 d5 2. c4 e6 3. Nf3 f5 4. Bg5 Nf6 5. Nc3 c6 6. e3 Bd6 7. c5 Bc7 8. Be2 Nbd7	Black	1673	1700

9. b4 OO 10. b5 Qe8 11. a4 Ne4 12. Nxe4 fxe4 13. Nd2 e5 14. Nb3 Qg6 15. h4 Qf7 16. OO h6 17. Bh5 Qe6 18. Bg4 Qg6 19. Be7 Rf7 20. Bh5 Qe6 21. Bxf7 + Qxf7 22. Bd6 Bxd6 23. cxd6 Qe6 24. dxe5 Qxe5 25. bxc6 bxc6 26. Nd2 Qxd6 27. Nxe4 1-0			
1. e4 e6 2. d4 d5 3. e5 f5 4. Nf3 g6 5. Nc3 Nh6 6. Bg5 Qd7 7. Qd2 Nf7 8. Bf4 c6 9. h3 Bg7 10. g4 a6 11. Ng5 h6 12. Nxf7 Qxf7 13. Bd3 g5 14. Bh2 f4 15. f3 Nd7 16. Ne2 Nf8 17. OOO Bd7 18. Qa5 Bc8 19. Bg1 Ng6 20. Bf2 OO 21. h4 Bd7 1-0	Black	1525	1500
1. e4 e5 2. Nf3 Nc6 3. d3 Bc5 4. a3 Nge7 5. h3 OO 6. Bg5 h6 7. Bxe7 Qxe7 8. Nc3 Nd4 9. Nxd4 exd4 10. Nd5 Qd6 11. Be2 c6 12. Nb4 Bxb4 + 13. axb4 Qxb4 + 14. Kf1 Qxb2 15. Kg1 Qb5 16. Kh2 Qe5 + 17. g3 d5 18. Bf3 dxe4 19. Bxe4 Qe6 20. Qf3 0-1	White	1336	1300

The server part of the application uses Express to exchange data with clients in JSON format. The server opens one of the ports for listening and downloads the files to send to the client. The Stockfish.js code is also running on the server. A separate engine object is created for each client. With each response of the engine, a line is generated, which indicates the depth of the search, position estimate, and subsequent moves, taking into account which the rating is set. At the end of the search, the engine offers a course according to the force specified in the parameters. To most accurately determine a player's rating, the engine plays the strongest way only when he loses significantly in position. Otherwise, he makes a move that is simply not a gross mistake. In both cases, in the presence of alternatives that do not significantly change the assessment of the position, the course will be chosen randomly.

4. Conclusion

As a result, we got a web application that uses an artificial neural network to calculate a chess player's rating after just one game with a computer. The parameters used are: the total value of the discrepancy between the quality of the player's moves and the moves calculated by the engine, the final position score and the number of moves.
Further research is related to the improvement of the project. Additional research is needed to determine the factors that affect the quality of a chess player's game and to determine the weights of these factors. Use a faster and more accurate engine or neural network, such as the AlphaZero neural network, to estimate the position. Optimize neural network code and SPA and update user interface for best user experience. Expand the range of platforms on which the application will be available.
This project can be reviewed by a source link: https://github.com/dendiod/elometer.

REFERENCES

1. FRUTOS-PASCUAL M., ZAPIRAIN B.: Review of the Use of AI Techniques in Serious Games: Decision Making and Machine Learning. IEEE Transactions on Computational Intelligence and AI in Games, 9 (2017) 2, 133-152.
2. RISI S., PREUSS M.: From Chess and Atari to StarCraft and Beyond. How Game AI is Driving the World of AI, (2020), 7–17.
3. OMRAN A., ABID Y., KADHIM H.: Design of artificial neural networks system for intelligent chessboard, 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS), 2017, 1-7.
4. VEČEK N., ČREPINŠEK M., MERNIK M., HRNČIČ D.: A comparison between different chess rating systems for ranking evolutionary algorithms, Federated Conference on Computer Science and IS, 2014, 511-518.
5. Webpage Elometer: *http://elometer.net*, 01.04.2020.
6. Webpage Lichess: *https://lichess.org/training*, 01.04.2020.
7. LYUBUN Z.: Fundamentals of the theory of neural networks, LNU Publishing Center named after Ivan Franko, Lviv, Ukraine, 2006, 72-73.
8. RASHID T.: We create a neural network. SPb, Alfa-kniga 2006.
9. Webpage: Github hyugit / stockfish-server - A minimal standalone server running stockfish.js: *https://github.com/hyugit/stockfish-server*, 04.04.2020.
10. Webpage: Github nmrugg/ stockfish.js - The Stockfish chess engine in Javascript: *https://github.com/nmrugg/stockfish.js*, 03.04.2020.
11. TYMOSHCHUK P.: Artificial neural networks. Lviv Polytechnic Publishing House, Lviv 2011, 20-22.