Scientific Research Publishing

Journal of Computer and Communications, 2018, 6, 1-12
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Application of Neural Network to Game Algorithm

Ying Tian[1], Sangkee Min[2*], Qinge Wu[3*]

[1]Key Laboratory for Road Construction Technology and Equipment of Ministry of Education, Chang'an University, Xi'an, China
[2]Department of Mechanical Engineering, Wisconsin University, Madison, WI, USA
[3]School of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou, China
Email: tianying@chd.edu.cn, *sangkee.min@wisc.edu, *wqe969699@163.com

How to cite this paper: Tian, Y., Min, S. and Wu, Q.E. (2018) Application of Neural Network to Game Algorithm. *Journal of Computer and Communications*, **6**, 1-12.
https://doi.org/10.4236/jcc.2018.62001

Received: November 29, 2017
Accepted: January 30, 2018
Published: February 2, 2018

Abstract

Intelligent was very important for command decision model, and it was also the key to improve the quality of simulation training and combat experiment. The decision-making content was more complex in the implementation of tasks and the nature of the problem was different, so the demand for intelligence was high. To solve better the problem, this paper presented a game method and established a game neural network model. The model had been successfully applied in the classification experiment of winning rate between chess game, which had good theoretical significance and application value.

Keywords

Game, Neural Network, Network Training, Pruning

1. Introduction

Today, simulation training and combat experiments are increasingly demanding on command decision models. Combat experiments need to solve the problem that the experimental credibility is not high and the space to think about is difficult to automatically explore. This poses a demand for the intelligence of the command decision model. In previous literatures [1] [2], a smart command decision model was solved all along. However, this paper will give an effective way to solve this problem by using the superiority of learning of neural network, and carry out the establishment of the game algorithm of neural network.

The modern power system in the sending, configuration, use and other aspects of the participants was increasingly diverse, and they had their own independent demands of interests, moreover, it will inevitably lead to conflicts of interest. Therefore, it was necessary to establish a fair and reasonable coordination

of interests and mechanisms of conflict resolved, balance and optimizing the interests of all parties [1]. The game theory could solve these problems. The game theory is a mathematical theory and method to study the phenomenon of struggling or competitive nature. The game theory considers the predictive behavior and actual behavior of individuals in the game and studies their optimization strategies.

When the energy with volatility and randomness was accessed to the grid in the wind power generation, photovoltaic power generation and others, in order to effectively control the risk to achieve better control effect, the random interference of nature such as wind, light and other uncertain factors could be seen as non-cooperative game side, then the above problems could be solved based on non-cooperative game theory [2]. The relationship between microgrid and large power grid had obvious game characteristics. It was of great practical significance to analyze the relationship between competition and cooperation by constructing the game model [3] [4].

Due to the need of confidentiality of communication, military and so on, and the environment for each signal was increasingly complex so that the characteristic information of the target had some ambiguity. However, fuzzy automata [2]-[8] are powerful tools to deal with fuzzy feature information. Based on this basis, this paper focuses on the establishment of target control system of fuzzy automata (FA). The system will be compared with the old method in the simulation. The simulation results show that its correct control rate is as high as 95.18%.

This paper will propose a game method, establish the game neural network model, and apply the model to the classification experiment of winning rate between chess.

2. Game Algorithm

Game was an important application of heuristic search all along. There have been several game systems as early as 1960s. One of the parties in the game tries to maximize the odds of winning goal, while the other tries to make the opposite side deviate from the winning goal. Both sides of the game and the other side are always moving to the most beneficial to their own state, on the other hand, that is, both sides are always moving to the most disadvantageous to each other's state.

Each step in the game is to achieve a Nash equilibrium. That is, the game with n Objects is described as $G=\{A,I,S,U\}$, where $A=\{1,2,\cdots,n\}$ is a set of all Objects that represent the behavior of the game coordination and the decision Object, whose purpose is to maximize own payment or utility level by selecting the strategy of action; I denotes that each Object owns information, including features of other Objects and strategy information of action; S represents a set of all possible strategies or actions of Object. All feasible strategy of an Object is called its strategy space, that is, S_i denotes the strategy space of the ith Object;

U is the income function, which indicates the benefits obtained by Object. It refers to the gains and losses of Object under the combination of the established strategy, that is, it is the level of benefits under a particular strategic combination. If the strategic combination $S^* = \left\{s_1^*, \cdots, s_i^*, \cdots, s_n^*\right\}$ in this issue is a Nash equilibrium, it must satisfy

$$U_i\left(s_i^*, s_{-i}^*\right) \geq U_i\left(s_i, s_{-i}^*\right) \quad \forall s_i \in S_i \tag{1}$$

where s_i^* indicates the strategy chosen by the ith Object; s_{-i}^* represents the vector that consists of strategies of all Objects except for the ith Object; U_i represents the benefit obtained by the ith Object; S_i represents the strategy space of the ith Object.

In a more complex game, according to a certain time and space costs, the search is extended to a certain layer so far. Because of the leaf node of the explicit sub-graph by this expansion is not the final success or failure state of the game, it cannot give the sure value that is the success or failure in the end. In this case, the values to the leaf nodes are assigned according to some heuristic functions that can show the probability of success or failure. Then, the value of each node in the search graph is pushed forward from bottom to top according to the reverse rule of the ∨/∧ method, including the root node. However, the value of this retreat to the root node does not indicate who will win, but only consider the limited number of steps that are described the number of layers of and/or in the search graph. The heuristic function value that corresponds to the best state in these layers can be achieved.

Each Object wants to win in the game. Therefore, the advantage of one side relative to the other side is directly estimated by some heuristic knowledge. In the chess, the entropy advantage is very important, so a simple heuristic strategy is always to calculate the advantageous difference of entropy between one side ∨ and the other side ∧, and maximizes the difference as far as possible. The more sophisticated some heuristic strategies assign different heuristic function values based on the differences of the entropy. The vast majority of games will have many heuristics information that can be easily used.

Give the heuristic function $h(n)$, and assume the benefit function is $U_i(n) = h(n)$, for example, $h(n)$ = ∨/∧. Then, the depth and width search method is used to find the optimal step according to the Nash equilibrium principle. In order to facilitate the discussion, the nine-grid game here is used to describe the game algorithm of the heuristic function.

Example 1. In the nine-grid game, assume one side ∨ is * side and the other side ∧ is the O side. Let ∨ start first.

The whole state in this issue is a total of 9! nodes. Even if the homogeneous chess game is removed, it is still a big number. Obviously, all blind searches here are not working. Therefore, the heuristic search method must be considered. In this case, the heuristic function $h(n)$ = ∨/∧ search method is used.

The whole row, the whole column or the whole diagonal of the chessboard are called the winning line. Here, the winning line method is defined as follows:

1) If there is no any chessman on a winning line, it is called a 0-order winning line. 0-order winning line can be regarded as belonging to the * side, can also be regarded as belonging to the O side, which they have no effect on the valuation.

2) If there is only one chessman of * (O) side on a winning line, it is called the first-order winning line of the * (O) side.

3) If there are two chessmen of * (O) side on a winning line, the winning line is called the second-order winning line of * (O) side.

4) If there are three chessmen of * (O) side on a winning line, the winning line is called the third-order winning line of * (O) side.

Thus, $h(n)$ can be defined as:

1) If the node n is a non-final node of the * side, the evaluation function of the * side is as follows:

$h(n)$ = (the number of first-order winning line of * side − the number of first-order winning line of O side) + 4 × (the number of second-order winning line of * side − the number of second-order winning line of O side) + a + 6 × (the number of third-order winning line of * side − the number of third-order winning line of O side) + b

where,

$$a=\begin{cases}+2 & \text{If the } * \text{ side takes the chessman, it can occupy the second-order winning line of the O side}\\ -2 & \text{If the O side takes the chessman, it can occupy the second-order winning line of the } * \text{ side}\\ 0 & \text{others}\end{cases}$$

$$b=\begin{cases}+3 & \text{If the } * \text{ side takes the chessman, it can occupy the third-order winning line of the O side}\\ -3 & \text{If the O side takes the chessman, it can occupy the third-order winning line of the } * \text{ side}\\ 0 & \text{others}\end{cases}$$

2) If the node n is the final node of the * side to win, then $h(n)=+\infty$.

3) If the node n is the failure final node of the * side, then $h(n)=-\infty$.

4) If the node n is a draw, then $h(n)=0$.

The search graphs of the algorithm ∨/∧ on the first and second steps can be obtained by using $h(n)$, as shown in **Figure 1** to **Figure 2**, respectively. The selected walking step is optimal and marked with a thick line.

Similarly, the search graph of heuristic algorithm ∨/∧ on the subsequent steps can be obtained.

From the search graphs with ∨/∧ of full two steps of the nine-grid game, both sides of the game are guided by $h(n)$ to carry out the search. The outcome is a draw, and then the mistakes of either side will be "self-defeated".

The heuristic function is important. If its definition is not appropriate, it may get undesirable results. In this case, the winning line is defined as: If there are only chessmen of * (O) side or empty on a winning line, but no chessmen of O (*) side, the winning line is called the winning line of the * (O) side. In this way, the heuristic function $h(n)$ of the * side can be defined as the evaluation function $h_1(n)$ as follows:

1) If the node n is a non-final node, then $h_1(n)$ = the number of winning lines of * side − the number of winning lines of O side.

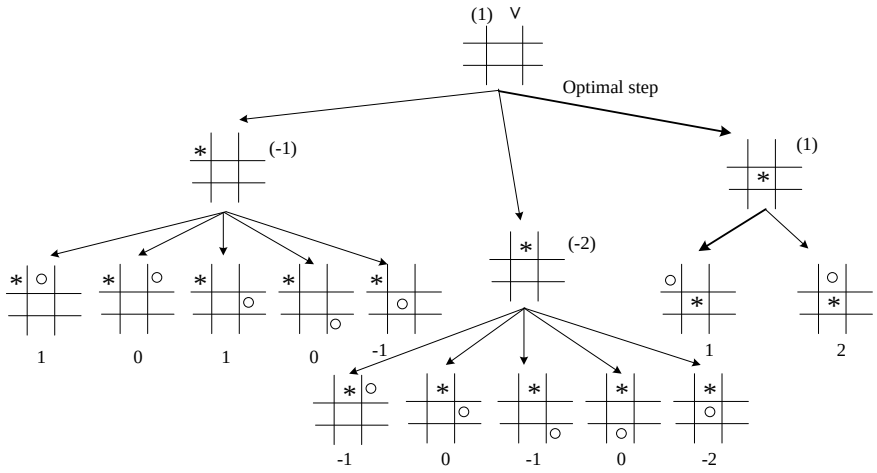

Figure 1. Search graph of the nine-grid game on the first-step.

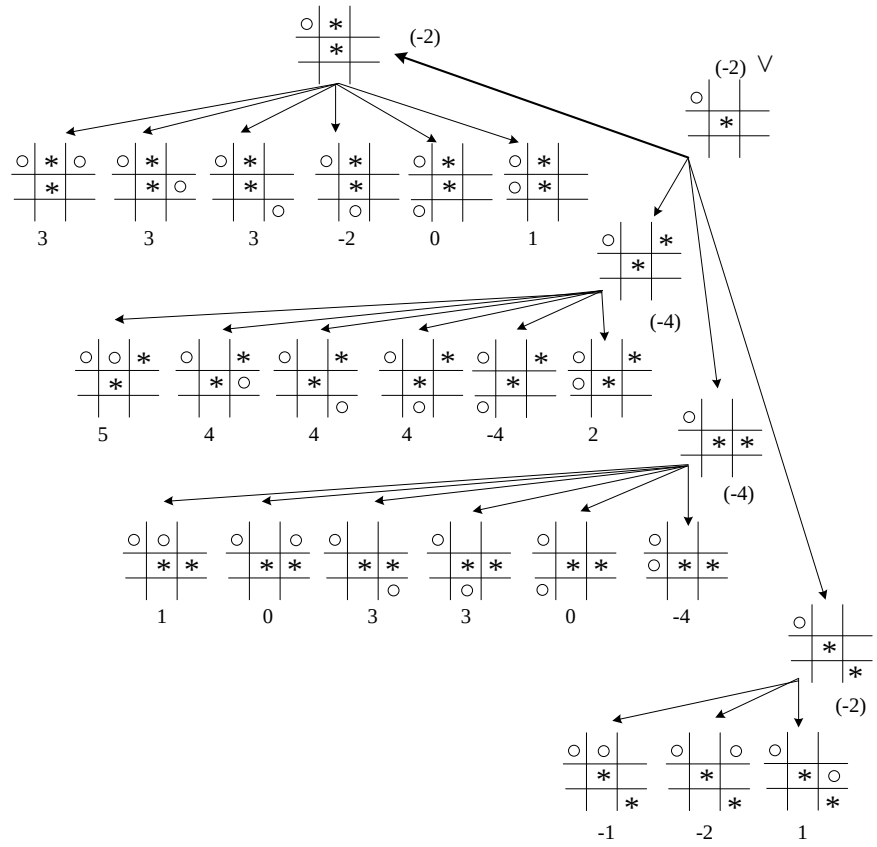

Figure 2. Search graph of the nine-grid game on the second-step.

2) If the node n is a draw, then $h_1(n)=0$.

3) If the node n is the final node of the * side to win, then $h_1(n)=+\infty$.

4) If the node n is the failure final node of the * side, then $h_1(n)=-\infty$.

Obviously, if the evaluation function $h_1'(n)$ is obtained by using the point of view of the O side to analyze the evaluation of the same chess, then there must be $h_1'(n)=-h_1(n)$.

The heuristic search graph of ∨/∧ on the first step can be obtained by using $h_1(n)$, which is the same as that of by using $h(n)$, as shown in **Figure 1**. The extended depth of the search graph is 2, which is a seemingly reasonable part of the game tree. The optimal step of both sides is marked with bold lines in this Figure.

However, the defect of $h_1(n)$ has been exposed in the search graph of ∨/∧ on the second step, because it is not accurate to guide the search in the chess game, as shown in **Figure 3**.

From **Figure 3**, it can be determined that the optimal step of the * side by the estimate of $h_1(n)$, as shown in **Figure 4(a)** and **Figure 4(b)**, because their evaluation is 1. The adverse step for the * side is shown in **Figure 4(c)** and **Figure 4(d)**, because they are evaluated as 0.

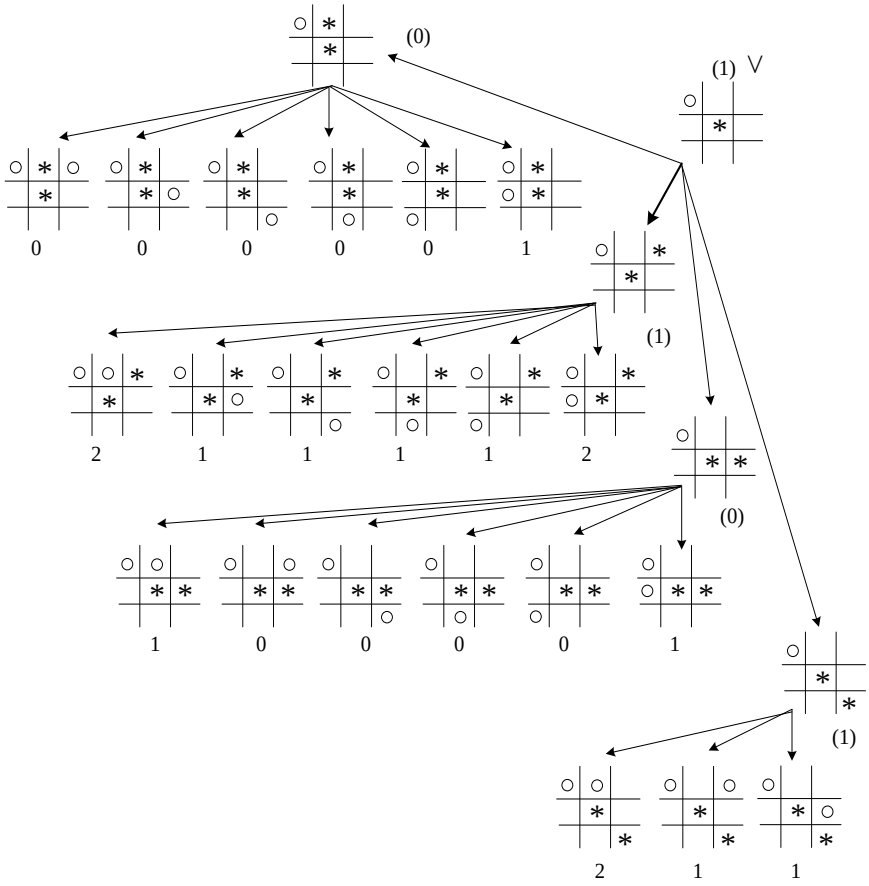

Figure 3. Search graph of the nine-grid game by using the second-step of $h_1(n)$.

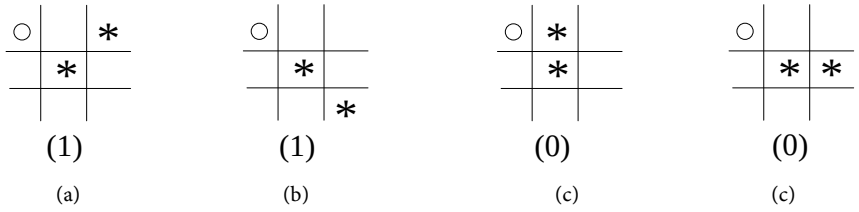

Figure 4. The possible choice of walking step by the * side.

This heuristic method with search ∨/∧ is to separate completely the process of generating the game tree from the process of calculating, evaluating and determining the optimal step. Only when the game trees with the specified depth all are generated is carried out, then starts to calculate, evaluate and determine the optimal step, this separation leads to lower search efficiency. If the calculation of the evaluation function of the endpoint and the pushing down operation of the inverted value of the intermediate node are completed at the same time as the tree grows, *i.e.*, while the game tree is generated and the evaluation is calculated at the same time, it is possible to reduce the workload of many generation and calculations. This technique is called ∨/∧ pruning technology.

3. Application of SOFM (Self-Organization Feature Mapping) Network in Classification of Chess Game

Learning the game algorithm by establishing a neural network is an important method in intelligent decision making. By analyzing the chess historical data of both sides of the game, the winning ratio of the two sides of the game is obtained in 100 games in a certain period of time, then the winning ratio can be defined:

$$r(x)=\frac{m}{n}$$

where $r(x)$ denotes the winning ratio of two sides; x denotes any one of both sides; m indicates the times of winning event; n indicates the total number of matches. The results regarding winning ratio between two sides are different, as shown in Table 1.

The data in the table is taken as the input sample P of the network. P is a two-dimensional random vector, and its distribution is shown in Figure 5.

The weight is trained by SOFM network. The distribution of the initial weights of the network is shown in Figure 6. Each point in Figure 6 represents a neuron, since the initial weights of the network are set to 0.5, so these points are coincident in Figure 6 and look like a point, actually 12 points.

W(i, 1) and W(i, 2) in Figure 6 are the coordinate of training weights of the ith neurons, respectively. Then the network is trained by using the training function. Assume the trained network can classify the input vector correctly. The number of training steps of network has great influence on network performance, so the number of steps is set to 100, 300, and 500, and the weight distribution is observed separately.

When the number of steps is 100, the distribution of weights is shown in Figure 7 (distribution of weights (the number of training steps: 100)), and the distribution of weights when the number of steps is 300 is shown in Figure 8 (distribution of weights (the number of training steps: 300)), the distribution of weights when the number of steps is 500 is shown in Figure 9 (distribution of weights (the number of training steps: 500)).

It can be seen from Figure 7 to Figure 9, the neurons begin to self-organize after the training of 100 steps is carried out, and each neuron can distinguish

Table 1. The winning ratio of both sides of the game.

Party A	0.5501	0.5113	0.5069	0.5001	0.6017	0.5298	0.5000	0.4961	0.5212	0.5011
Party B	0.4499	0.4887	0.4931	0.4999	0.3983	0.4702	0.5000	0.5039	0.4788	0.4989

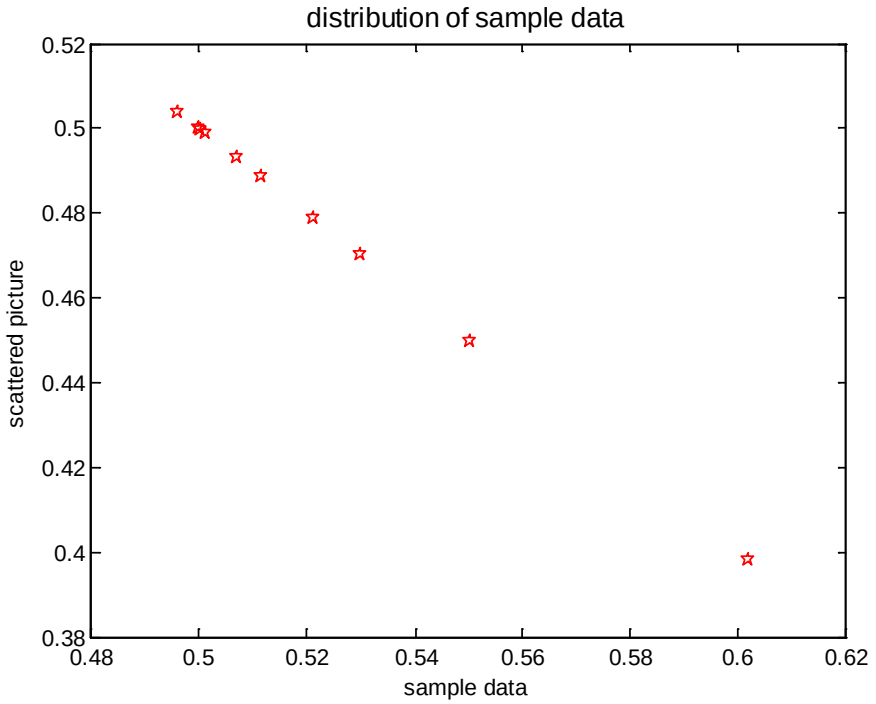

Figure 5. Distribution of sample data.

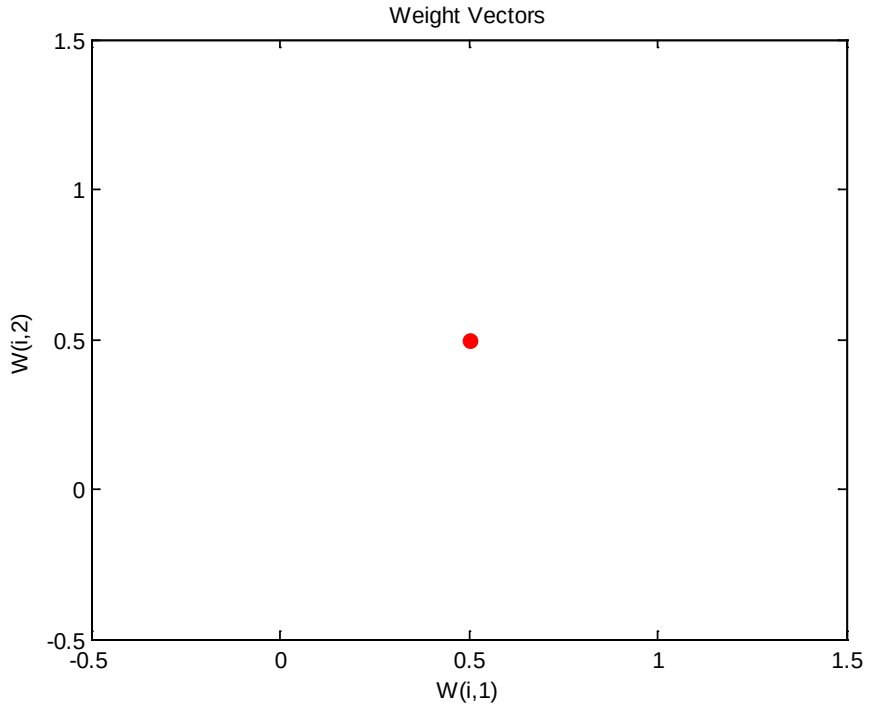

Figure 6. Distribution of initial weights of network.

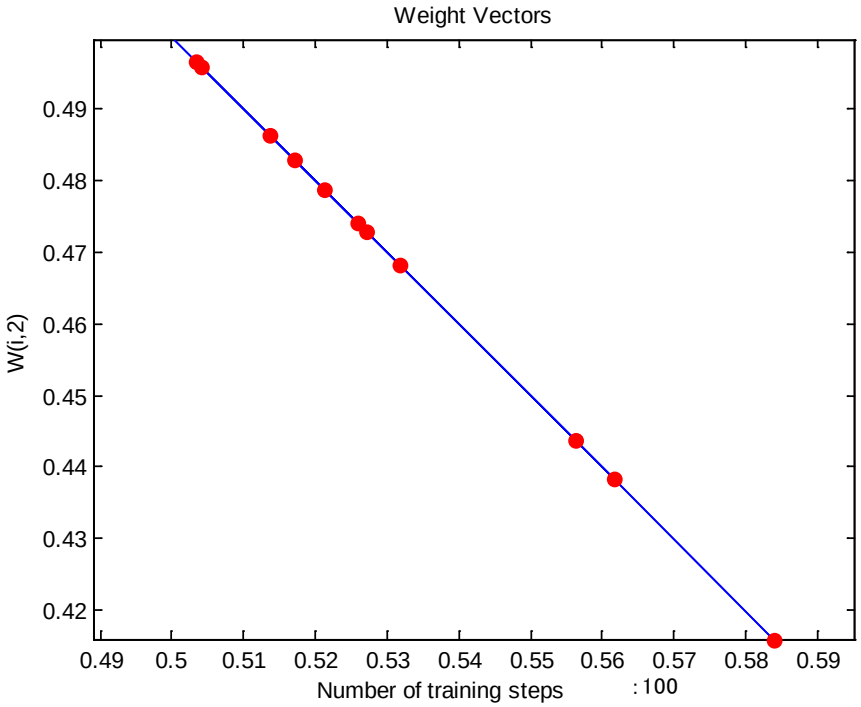

Figure 7. Distribution of weights when the number of steps is 100.

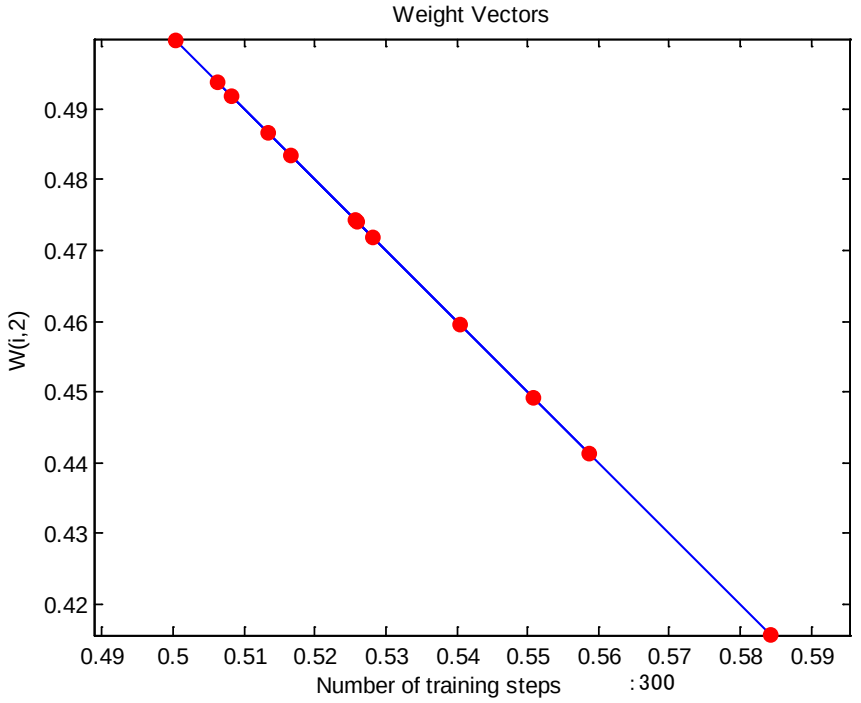

Figure 8. Distribution of weights with 300 steps.

different samples. With the increase of the number of training steps, the distribution of neurons is more reasonable. However, when the number of times of training reaches a certain value, the change of the distribution of weights is not obvious, because the weights are out of range the neighbor of the winning

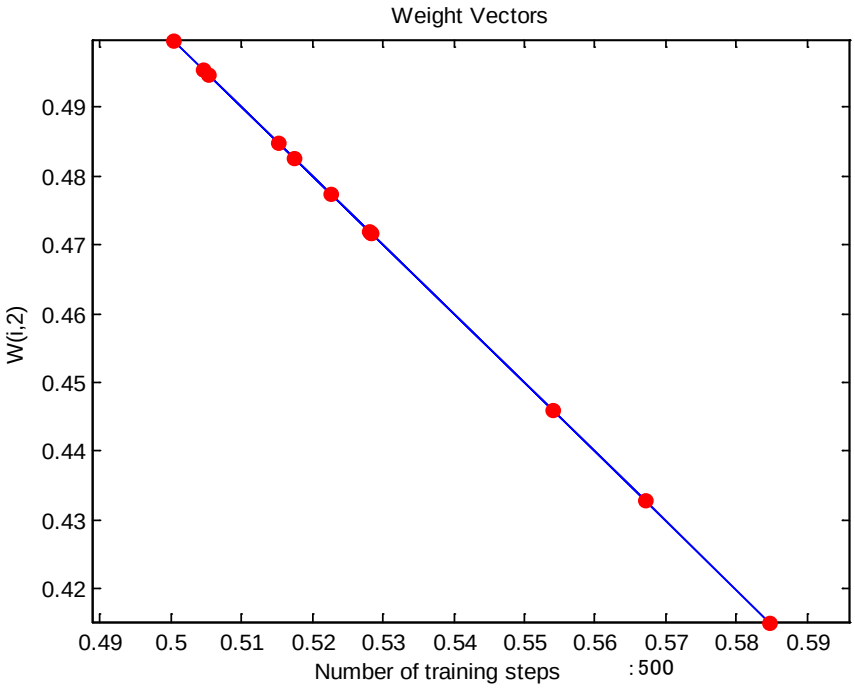

Figure 9. Distribution of weights when the number of steps is 500.

output $N(i^*)$ which is a near neighbor of the winning output neuron i^*, and which is designated by the distance between the output neurons. For example, the distribution of weights after training 300 steps and that of training 500 steps are similar. The adjustment of weights is given as follows: For each output neuron $i \in \{N(i^*), i^*\}$, the weight is updated by the expression $\omega_{kj}(t+1) = \omega_{kj}(t) + \eta(t) d_k(t)$, if $j \in N(i)$. where d_k is the difference of weights of outputs between the present time and last time, and the $\eta(t) = \eta$ has been determined in experiment experience. This rule only updates the near neighbor of the winning output neuron.

After the end of the network training, the weights are determined. A value at each time is input, the network will automatically classify it. Therefore, the network can be tested by using this feature. First, the sample vector P is input to the network for testing, and the simulation function is used to observe the classification of the sample data by network. The simulation results are Output = [3, 9, 9, 12, 1, 5, 12, 12, 10, 12]. The topology of neural network after training is shown in **Figure 10**.

Now, the winning ratio p = [0.5; 0.5] for both parties in a certain period of time is input to verify which category it belongs to. The simulation result is Output = 12. This shows that the 12th neuron of the network is stimulated at this time, so p belongs to the fourth category. By comparing the data directly, p is indeed very close to the data in group 4, group 7 and group 10 of samples.

4. Conclusion

As a classical method to analyze the benefit relationship between the multi-

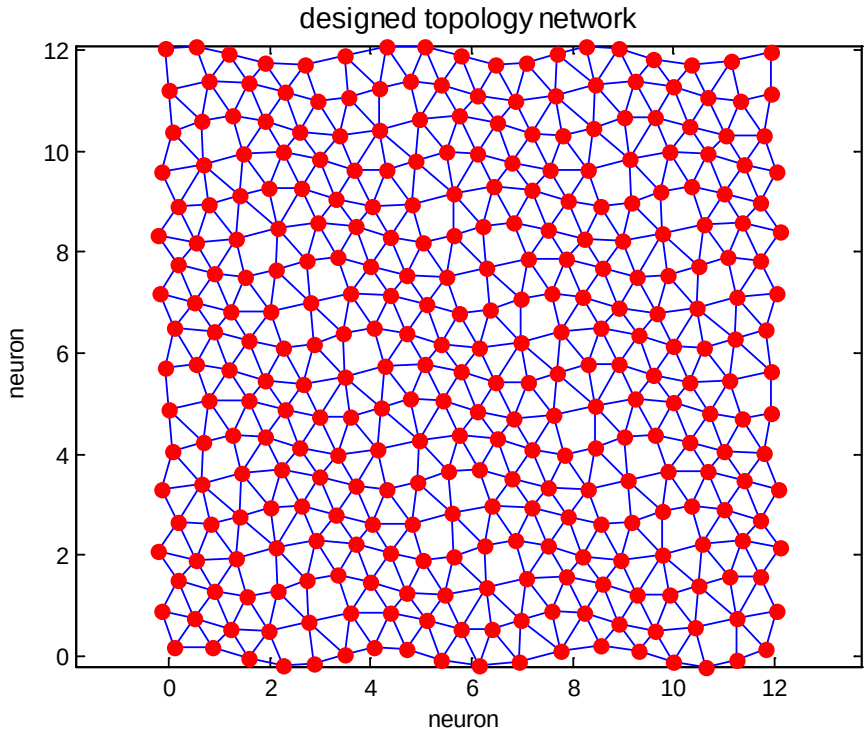

Figure 10. Designed topology of network.

decision main parts, the game theory is widely used in all aspects of macro-decision-making strategy and micro-decision-making system. In this paper, a smart command decision model was solved by using the superiority of learning of neural network, and the establishment of the game algorithm of neural network was also given. In general, the game theory plays an increasingly important role in the application of field of engineering decision-making research from macro to micro, from qualitative to quantitative. With the rise of the concept of Internet with various decision-making, the democracy and fairness of decision-making will be paid more and more attention, and the game method on neural network is a powerful tool to solve the above problems.

Acknowledgements

This work is supported by National 973 Program (No. 613237); Henan Province Outstanding Youth on Science and Technology Innovation (No. 164100510017); Natural Science Basic Research Plan in Shaanxi Province of China with Grant (No. 2014JQ7248), respectively.

References

[1] Li, Y., Wei, L.J., Chi, Y.N., Liu, C. and Zhang, Z.K. (2016) Research on Reactive Voltage Characteristics and Control Strategy of Offshore Wind Farm. 12*th IET International Conference on AC and DC Power Transmission*, *Power*, *Energy*, & *Industry Applications*, Beijing, 28-29 May 2016, 1-6.

[2] Mei, S.W., Zhang, D., Wang, Y.Y., *et al.* (2014) Robust Optimization of Static Re-

serve Planning with Large Scale Integration of Wind Power: A Game Theoretic Approach. *IEEE Transactions on Sustainable Energy*, **5**, 535-545. https://doi.org/10.1109/TSTE.2014.2299827

[3] Zhang, J.H., Ma, L. and Liu, N. (2016) Application and Prospect of Game Theory in Microgrid. *Electric Power Construction*, **37**, 55-61.

[4] Senior, J., Stevens, J. and Smith, C.A. (2017) The Importance of Engineers in Successful Consenting for Project Developers. 13*th IET International Conference on AC and DC Power Transmission*, *Power*, *Energy*, & *Industry Applications*, Manchester, 14-16 February 2017, 1-9. https://doi.org/10.1049/cp.2017.0001

[5] Lukas, E., Mölls, S. and Welling, A. (2016) Venture Capital, Staged Financing and Optimal Funding Policies under Uncertain. *European Journal of Operational Research*, **250**, 305-313. https://doi.org/10.1016/j.ejor.2015.10.051

[6] Cui, G.H., Li, M.C., Wang, Z., *et al.* (2015) Analysis and Evaluation of Incentive Mechanisms in P2P Networks: A Spatial Evolutionary Game Theory Perspective. *Concurrency and Computation Practice Experience*, **27**, 3044-3064. https://doi.org/10.1002/cpe.3207

[7] Cheng, D.Z., He, F.H., Qi, H.S., *et al.* (2015) Modeling Analysis and Control of Networked Evolutionary Games. *IEEE Transaction on Automatic Control*, **60**, 2442-2455. https://doi.org/10.1109/TAC.2015.2404471

[8] Zhang, C., Pan, R., Abhijit, C., *et al.* (2014) Effect of Security Investment on Evolutionary Games. *Journal of Information Science and Engineering*, **30**, 1695-1718.