World Journal of Innovation and Modern Technology, Vol. 8, Issue 5, (May) 2025
ISSN 2682-5910
DOI: 10.53469/wjimt.2025.08(06).01

Research on a Software Testing Method based on BP Neural Network

Tingting Luo[1], Miaofen Feng[2], Yizhen Wan[3], Binbin Song[3]

[1]Nanchang Health Vocational and Technical College
[2]Jiangxi Teachers College
[3]SPIC JiangXi Electric Power CO., LTD. New Energy Power Generation Branch.

Abstract: ***Software will have various problems in the process of writing and designing, and software testing is to find out as many defects and potential errors as possible through different test cases, so as to effectively ensure the quality of software. However, the current software testing is still mainly based on manual testing, which is difficult to adapt to the needs of modern software development. Especially in the military industry, the safety and reliability of software is particularly important. How to use as comprehensive test cases as possible to find out more potential errors. In this research paper, a software testing method based on BP neural network is proposed. In the BP neural network training as perfect as possible test case model, accurate and fast discovery of software errors, thereby improving the efficiency of software testing. At the same time, when the modified part of the code is to be retested, the test case model of BP neural network can test the other errors caused by the code change, thus effectively improving the work efficiency.***

Keywords: Software testing, Test cases, BP neural networks, Testing methods.

1. INTRODUCTION

Internet development in the 21st century is particularly rapid, mobile communication products have entered into all aspects of life, software products have become an indispensable part of human life. With the development of the market economy, the market demand for high-tech software products is continuing to increase, Internet users and mobile users is gradually accelerating the scale of growth, resulting in a significant increase in the software market demand for the product, which also requires a significant reduction in the time needed for software testing. In order to control the time cost of software testing, testers have developed a relatively large number of testing tools through technical means. For the same type of software, the development of the code used or the logical order of the code are similar, thus making the software architecture tends to be relatively stable, and the relatively stable software architecture gives support to BP neural network testing. But only these are not enough, want to obtain a stable BP neural network test model, first of all to obtain a large number of related aspects of the data, and these data can not be used directly as a test model, these data need to be screened by professionals to identify, select the representative or valuable data can be used as a training set. Software testing methods based on BP neural networks will greatly improve testing efficiency, save testing time and cost, and improve software testing coverage [1].

2. SOFTWARE TESTING

Software testing is a method to find out the defects of software products in the process of requirement analysis and software design, and to check whether the developed products satisfy the user requirements, business requirements, functional requirements and non-functional requirements, so as to ensure the quality of the software better [2].

Software testing work from the requirements analysis can be carried out on the test link, until the end of the project, essentially involved in the entire project life cycle. Software testing methods mainly cover black box testing, grey box testing and white box testing. In our current military products in the use of the most for the black box test, black box test, also known as functional testing, test the product function to meet the demand requirements. The principle is to consider the programme as a closed black box, without considering its internal structure [3]. And a large amount of work in software testing is basically done by manual testing, which takes more time and cost. How to reduce the large amount of repetitive and tedious work is a worthy research topic. Of course we can introduce automation, but it is basically required to run a stable system under the premise that if the current system quality is not good enough, the use of automated testing does not effectively improve the efficiency of the work. How to seek a better method to solve the current problems, BP neural network is a good software testing method research.

3. BP NEURAL NETWORK

3.1 Principle of BP Neural Network

BP Neural Network (Backpropagation Neural Network, BPNN) is a multi-layer feed-forward neural network trained on the basis of the error backpropagation algorithm, whose core mechanism is composed of two complementary processes, namely, forward propagation of signals and backpropagation of errors. The core principle of the mechanism is composed of two complementary processes: forward propagation and back propagation. In the forward propagation phase, the pre-processed data set is injected from the input layer, and feature mapping is performed sequentially through hidden layers containing single or multiple layers, and finally the prediction results are generated from the output layer [4]. At this time, the initial weights and bias parameters of each layer of the network are randomly initialised, and the deviation of the output results from the true value is calculated by the error analysis module, and then enters into the back propagation link, i.e., the error is passed from the output layer layer by layer, and the weights and bias parameters of each layer are iteratively updated to make the output value approach the true value step by step. When the mean square error of the output layer is less than a preset threshold, the training is terminated, and the optimal weights and bias parameters determined at this time will make the network model stable, forming a software test model that can be used for practical tasks [5].The typical architecture of BP neural network contains input layer, hidden layer (which can contain single layer or multi-layer), as well as the output layer, and the signal transmission is achieved through the connection of the weights of the various layers, and the structure of the network is shown in Figure 1. The network structure is shown in Figure 1.

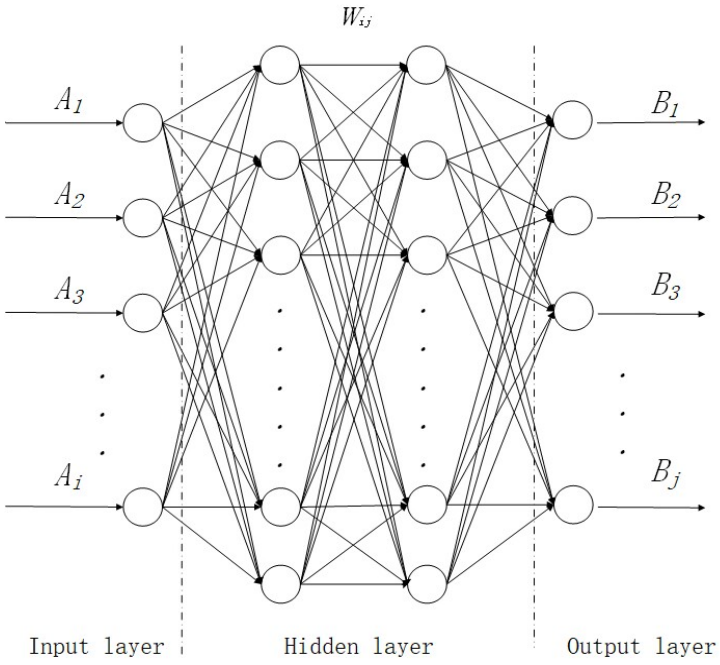

Figure 1: BP neural network

$A_i(i = 1,2,\ldots,n)$ represents the set of input signals, w_{ij} represents the weight values from neuron i to neuron j, and $B_j(j = 1,2,\ldots,n)$ represents the set of output signals from neuron.

3.2 Training Process Using BP Neural Networks

3.2.1 Forward propagation process

1) The screened dataset is firstly input into the BP neural network, and the input signal value of each unit is multiplied by the weight value of the item one by one and summed, followed by the introduction of bias parameter for correction, and finally, the output signal of the current neuron is produced after nonlinear transformation. During this forward propagation process, the weight values and bias parameters of the internal layers are assigned using a random initialisation strategy, and the randomly generated weight values and bias parameters are dynamically adjusted and updated for optimisation during the error back propagation process until the model is

trained to a stable state. This basic constituent unit is called a neural node, and each layer contains weight values and bias parameters. The formula is as follows:

$$I_j = \sum_i W_{ij} O_i + \Theta_j \quad (1)$$

In the expression, O_i denotes the output of the first i neuron; W_{ij} denotes the weight value from neuron i to neuron j; Θ_j denotes that it is the bias parameter of neuron j; and I_j denotes that it is the net input value of the first j neuron.

A nonlinear transformation is required before obtaining the output value and the nonlinear excitation function is a sigmoid function. The formula is given below:

$$O_j = \frac{1}{1+e^{-I_j}} \quad (2)$$

In the expression, O_j denotes the output of the neuron j; the rest of the variables in Eq. are detailed in Eq. (1). As in the multilayer neural network architecture, the output of each layer of neurons is used as the input of the next layer.

3.2.2 Error back propagation process

Error analysis is performed between the output results obtained in the forward propagation and the true values, and the error is passed backward from the output layer to the input layer layer by layer through each hidden layer, thus updating the weight values and bias parameters in the network. The error value in the output layer, Eq:

$$E_j = O_j(1 - O_j)(T_j - O_j) \quad (3)$$

In the expression, E_j denotes the error value of the output layer; T_j is the true value; the rest of the variables in the equation are detailed in equation (2).

After the error value is computed in the final output layer, the error is back-propagated layer by layer through the hidden layers to the input layer. The error of each layer is used to calculate the error of the previous layer, similar to dominoes, until it returns to the top layer where the error calculation is completed. By this iterative approach, an estimate of the error in each layer of the neural network can be obtained. The formula is as follows:

$$E_j = O_j(1 - O_j)\sum_k E_k W_{jk} \quad (4)$$

In the expression, E_k indicates the error of the previous layer; $\sum_k E_k W_{jk}$ indicates that each error value of the previous layer is multiplied with the corresponding connected weights, and then all these products are accumulated; the rest of the variables in the formula are described in Eqs. (1) to (3).

The basic idea of BP neural network is gradient descent method to update the weight values, by using the output error of the neuron$j$ and the input signal of the neuron$i$ and the set learning rate to update the weight values, the continuous iterative parameter adjustment process, so that the error between the network output and the real target is gradually reduced until it meets the pre-set convergence conditions, and realises the optimal configuration of the weight values. The formula is as follows:

$$\Delta W_{ij} = (l) E_j O_i \quad (5)$$

$$W_{ij} = W_{ij} + \Delta W_{ij} \quad (6)$$

In the expression, ΔW_{ij} denotes the update weight value; l denotes the learning rate; the rest of the variables in Eq. are detailed in Eqs. (1)(3). In model training, the weight update value is achieved by superimposing the current value of weights with the update amount. The learning rate, as a core parameter of gradient descent, takes the value in the range (0,1) and directly regulates the speed of the cycle. In practice, an adaptive regulation strategy is often used: a larger learning rate is given at the beginning to accelerate the exploration, and gradually decayed at a later stage to fine-tune the parameters. In fact, it is relative when setting larger values in the first few times, because setting larger is easy to miss the lowest point. Therefore, the smaller learning rate is used at special times, so that the more fine-tuned it can be, it is obvious that the speed will drop accordingly. In the actual processing of the model is basically non-linear, learning rate settings directly affect the gradient descent algorithm performance. Learning rate setting is too large, easy to span too long, will come back and forth to form the ups and downs of the shock, on the contrary, the setting is too small, although it will be easier to stabilise, but the time consumed is longer. If the training is not optimised, the optimal learning rate cannot be obtained. The optimal learning rate can

only be obtained by constant dynamic adjustment in actual training. This paper proposes an adaptive learning rate adjustment algorithm based on error feedback: when the value of the loss function decreases, the learning rate increases by 3% to accelerate convergence; if it rises, it is rolled back to the previous round's value and scaled down by 50%, so as to balance the training efficiency and optimisation accuracy.

The bias parameters are updated using the gradient descent law, with the help of the output error and learning rate on the neuron$j$. The formula is given below:

$$\Delta\Theta_j = (l)E_j \tag{7}$$

$$\Theta_j = \Theta_j + \Delta\Theta_j \tag{8}$$

In the expression, $\Delta\Theta_j$ denotes the update of the bias parameter; the rest of the variables in the equation are detailed in Eqs. (1) (3) (5). The bias parameter update value is achieved by superimposing the current value of the bias parameter with the update amount.

In the construction of BP neural network, the error back propagation process is the core link, and the gradient descent method is especially critical. This method is based on the gradient search technique to adjust the weight values and bias parameters layer by layer during the back propagation process, so that the error value is gradually reduced. When the error sum of squares between the actual output of the network and the real value reaches the minimum, the training process is terminated, and the optimised weight values and bias parameters of each layer are saved, so that a software testing network model with stable generalisation ability can be constructed.

4. SOFTWARE TESTING METHODS USING BP NEURAL NETWORKS

The pre-processed dataset is fed into the BP neural network to carry out the training task. During the training period, the weights and bias parameters of each layer of the network are updated and optimised in real time according to the backpropagation algorithm. When the sum of squares of the errors in the output layer reaches the global minimum during the error back propagation process, the training process is terminated, and the optimal weights and bias parameters of each layer are saved, and the constructed network model can be regarded as a software test model with stable prediction performance; if the predetermined error threshold is not reached, then iterative training is continued [6][7]. After completing the training, the resulting model is verified by practical testing, and if the test results are highly consistent with the manual test results, it indicates that the model has practical application value and can be put into use.

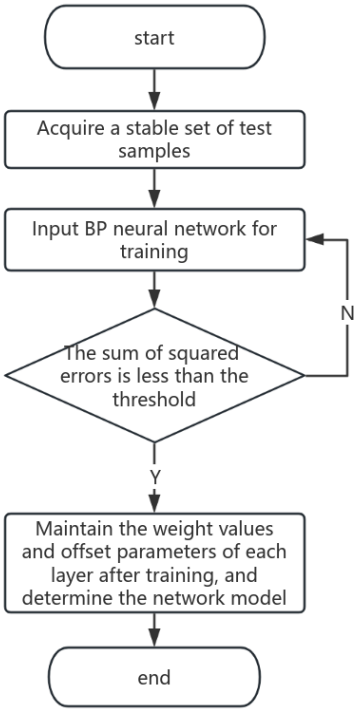

Figure 2: WiFi Calibration Flowchart

When the product is obtained, the software testing model is used to test the software project directly, to find out the problems of the software, while the current testing model is still in the primary stage, it can be appropriate to use manual verification, BP neural network in the process of working is also in the process of independent learning, the more times it is used, that is, the richer the sample set, the more stable and reliable the testing model tends to be. When the test bugs are identified, submitted to the developer to continue to modify the code will be changed, this time the code is different from the previous code, in order to prevent the developer in the bug modification will affect other normal functions. A new round of testing is also needed to start from scratch, the use of test models for software testing, but also can greatly reduce the repetitive workload, improve the efficiency of software testers, while on a certain basis the use of machine automated testing to ensure that the high quality of the software [8][9].

5. CONCLUDING REMARKS

The software testing method based on BP neural network proposed in this paper can theoretically meet the needs of modern software development. Firstly, a sufficiently stable software test sample is constructed, and then the pre-processed sample set is trained with the help of BP neural networks to obtain a stable and optimal nonlinear calibration test model, which is finally used to test the software, which can find out the potential errors and improve the quality of the software. In this paper, the BP neural network-based software testing model still needs to be improved through experiments so that it can become a usable model, and at the same time, obtaining the training set is also a tricky matter, and the next step needs to solve the above problems.

REFERENCES

[1] TAO Juan, ZOU Hongbo, ZHOU Dong. Short-term load forecasting model based on boosted artificial neural network [J]. Electrotechnical Materials, 2021(02): 53-56.
[2] Lameu Ewandson L., Borges Fernando S., Iarosz Kelly C., Protachevicz Paulo R., Antonopoulos Chris G., Macau Elbert E.N., Batista Antonio M.. Short-term and spike-timing-dependent plasticity facilitate the formation of modular neural networks[J]. Communications in Nonlinear Science and Numerical Simulation, 2021, 96(prepublished).
[3] Chadha Gavneet Singh, Panambilly Ambarish, Schwung Andreas, Ding Steven X. Bidirectional deep recurrent neural networks for process fault classification. [J]. ISA transactions, 2020, 106.
[4] Junxuan Wang, Yinan Wang. 5G indoor joint positioning algorithm based on deep neural network[J]. Journal of Xi'an University of Posts and Telecommunications, 2020, 25(04): 43-47.
[5] Feng Xiaobin. Design and implementation of automated testing framework for enterprise point software [D]. Southeast University, 2019.
[6] Deng Shaowei. Application of software automation testing methods[J]. Electronic Technology and Software Engineering, 2019(21): 32-33.
[7] Shi Yingchao. Discussion on computer software testing technology and development application[J]. Information and Computer (Theory Edition), 2019, 31(21): 88-89+92.
[8] Xu Z. Research on software defect prediction methods for different scenarios [D]. Wuhan University, 2019.
[9] Wang Zubing, Sun Ning. Analysis of the problems and countermeasure suggestions of CBB management work carried out by military electronic product enterprises[J]. China standardisation, 2019(15): 73-76.