Computer Science and Information Systems 16(3):773–795 https://doi.org/10.2298/CSIS181001026D

Design of Intrusion Detection System Based on Improved ABC_elite and BP Neural Networks

Letian Duan, Dezhi Han, and Qiuting Tian

College of Information Engineering, Shanghai Maritime University
Shanghai 201306, China
1812796760@qq.com, dzhan@shmtu.edu.cn, tqt2018@163.com

Abstract. Intrusion detection is a hot topic in network security. This paper proposes an intrusion detection method based on improved artificial bee colony algorithm with elite-guided search equations (IABC_elite) and Backprogation (BP) neural networks. The IABC_elite algorithm is based on the depth first search framework and the elite-guided search equations, which enhance the exploitation ability of artificial bee colony algorithm and accelerate the convergence. The IABC_elite algorithm is used to optimize the initial weight and threshold value of the BP neural networks, avoiding the BP neural networks falling into a local optimum during the training process and improving the training speed. In this paper, the BP neural networks optimized by IABC_elite algorithm is applied to intrusion detection. The simulation on the NSL-KDD dataset shows that the intrusion detection system based on the IABC_elite algorithm and the BP neural networks has good classification and high intrusion detection ability.

Keywords: Intrusion Detection, Machine Learning, BP Neural Networks, Improved ABC_elite.

1. Introduction

With the development of network technology, especially the widespread use of the Internet, it is more convenient to interconnect and interoperate. However, the problem of network security is also becoming increasingly prominent, and lawbreakers may seek benefits through damage to the network. Therefore, the detection and defense of network attacks is a hot topic of network security. Attackers usually leverage network protocol vulnerability to perform network attacks, including Denial of Service (DoS), User to Root (U2R), Probe and Root to Local (R2L). So far, methods for detecting network attacks include classification and clustering, which detect network attacks by analyzing network data flow.

Machine learning methods have been widely used to identify different types of attacks, and machine learning methods can help network administrators take appropriate measures to deal with network attacks. However, most of these traditional machine learning methods belong to shallow learning and require a lot of feature extraction and feature selection. They can't solve the problem of a large number of attacks and intrusion data classification problems in real network environments. In addition, shallow learning is not suitable for the intelligent analysis and forecasting of high-dimensional and massive data. However, the Backpropagation (BP) neural network model has good adaptability, self-learning ability and nonlinear approximation ability, can meet these requirements, and

has been widely used in prediction, modeling, classification, adaptive control and other fields.

The Swarm intelligence is simply defined as the collective behavior of decentralized and self-organizing swarms. As we all know, these swarms can be birds, fish and the colony of social insects such as ants and bees. In the 1990s, especially two methods, based on the ant colony and bee colony, attracted the interest of researchers. The Swarm intelligence requires a colony to meet self-organizing features. Since the 21st century, researchers have started to be interested in new intelligent methods using bee colony behavior. In the past decade, some algorithms based on various intelligent behaviors of honey bee colony have been proposed. The population algorithm originates from the resolution of numerical optimization problems and is a meta-heuristic target optimization algorithm.

In view of the shortcomings of traditional neural network training methods, researchers began to try to apply heuristic search algorithms to artificial neural network design and parameter optimization. That is to say, heuristic search methods can be used to train neural networks. The fusion of the heuristic search algorithms and neural networks are called evolutionary neural networks [29,30]. Artificial Bee Colony Algorithm (ABC) is a new heuristic search algorithm. It was proposed at Erciyes University in 2005. The algorithm is derived from the research and simulation of honey bee collective behavior. Compared with other heuristic search algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Differential Evolution (DE), ABC algorithm is simple, robust as well as need few parameters.

The optimization technology is an application technology that uses mathematics as the basis to find the optimal feasible solution for the objective optimization problem under certain conditional constraints. Before the 1980s, the optimization algorithms mainly were used to mathematical analysis, iterative solution and other methods to solve practical problems. They were called traditional optimization algorithms. These methods have complete theoretical analysis and mathematical proofs, and have also achieved good results in optimization problems such as continuous and low-dimension. But they seem to be powerless for multi-peak, high-dimension and discontinuous optimization problems. From the 1980s, some novel heuristic search algorithms emerge, which are different from traditional optimization algorithms. For example, the Genetic Algorithm is a computational model that simulates the natural selection and genetic mechanism of Darwin biological evolution. The idea of Simulated Annealing algorithm comes from the process of solid material annealing in physics. All these algorithms simulate some processes that occur in nature.

The remainder of this paper is organized as follows. In Section 2, we review the related work about the development of artificial bee colony algorithm, as well as using artificial bee colony algorithm to train the artificial neural network, and some intrusion detection methods in recent years. And we show the innovation about our intrusion detection method, which is different from other intrusion detection methods. In Section 3, the description of BP neural networks about neuron model, multilayer feedforward neural networks, and Backpropagation algorithm is introduced. In Section 4, we describe the artificial bee colony algorithm and propose the improved ABC_elite algorithm. In Section 5, we introduce the BP neural networks based on IABC_elite algorithm. Section 6 highlights the experiment about the dataset description, the data preprocessing, as well as the

experiment results. We analyze the experiment results in detail. Finally, the conclusion is discussed in Section 7.

2. Related Work

The artificial bee colony algorithm is a heuristic swarm intelligence algorithm designed by Karaboga in 2005 to mimic the collective behavior of the honey bee. It was originally designed to solve some numerical optimization problems [12]. The artificial bee colony algorithm was used to optimize multivariate functions, and compared with algorithms such as genetic algorithm (GA) and particle swarm optimization (PSO). The results show that ABC is superior to other algorithms [14]. However, the artificial bee colony algorithm is good at exploring the solution, but it is poor in exploitation and easy to fall into a local optimum. Ref. [32] proposed an improved ABC algorithm called gbest-guided artificial bee colony (GABC) algorithm, which combines the information of the global optimal solution into the solution search equation to improve the exploitation. Ref. [27] proposed a multi-strategy ensemble artificial bee colony (MEABC) algorithm. In MEABC, a pool of distinct solution search strategies coexist throughout the search process and compete to produce offspring. The MEABC method effectively improves the performance of ABC on continuous optimization problems. Ref. [2] proposed an artificial bee colony algorithm with Elite-Guided Search Equations combined with a depth-first framework, named DFSABC_elite. Prioritizing more computing resources for better solutions enhances the exploitation capabilities of the algorithm.

The use of artificial bee colony algorithm to train artificial neural networks also has a certain historical background. In Ref. [13], the artificial bee colony algorithm is used to train neural networks to solve the benchmark problems of XOR, 3-bit parity, 4-bit encoder-decoder, etc., which embodies the ability of artificial bee colony algorithm to train neural networks. Ref. [21] proposed a hybrid algorithm combining artificial bee colony algorithm and Levenberq-Marquardt algorithm (ABC-LM) to train artificial neural networks. The neural network is first trained with ABC, and then LM continues to train the neural network using the optimal weight set of the ABC algorithm and attempts to minimize training errors. The results of the experiments show that the hybrid ABC-LM algorithm has better performance. However, this method has only been tested on benchmark issues such as XOR, 3-bit parity, 4-bit encoder-decoder, etc., and has not been tested on high dimensional classification benchmark problems.

In 1999, Wenke et al. applied the machine learning method to intrusion detection for the first time. By analyzing the network data traffic, an anomaly detection model was obtained [18]. In Ref. [23], a distributed denial of service attack (DDos) detection method based on BP neural network is proposed, which is carried out by simple traffic classification and feature selection. The detection of DDos traffic reaches a certain detection rate. However, this method has a long training time and is easy to falling into a local optimum. Ref. [31] proposed a recurrent neural network (RNN) deep learning method for intrusion detection, which was simulated in binary-classification and multi-classification environments, but the method still needs to improve the detection accuracy and reduce the training time of the neural network model.

This paper studies the intrusion detection method based on the improved ABC_elite and BP neural networks. The IABC_elite algorithm proposed in this paper improves the

DFSABC_elite algorithm by integrating the Gaussian search equation in Ref. [19] into the DFSABC_elite algorithm, which improves the exploitation ability of the artificial bee colony algorithm and accelerates the search for the optimal solution. The use of IABC_elite to optimize the BP neural network avoids the neural network falling into local optimum and solves the problem of slow convergence of neural network training to some extent. The intrusion detection method based on IABC_elite and BP neural network improves the accuracy of intrusion detection and reduces the false positive rate. In the NSL-KDD dataset, KDDTrain+ and KDDTest-21 were respectively selected as training dataset and testing dataset to verify the performance of the algorithm.

3. BP Neural Networks

The Artificial Neural Network is an intelligent algorithm by simulating the structure of the neural system and information delivery of biological neural networks. Each single neuron performs simple operations. When numerous neurons are combined and operated together, they make up a huge distributed and parallel computing model.

3.1. Neuron Model

Neurons are basic information processing units for neural networks. Fig. 1 shows a model of a neuron which is the basic unit of artificial neural networks. The neuron model comprises three basic elements:

1) Synapses, each of them are characterized by its weight value. In particular, the synaptic weight value w_{ij} is a coefficient that multiplies the input signal x_i. The first subscript of the weight value refers to the query neuron, and the second subscript refers to the receptor neuron where the weight value is located.

2) A linear combiner, it is for summing the input signals multiplied by the weight value of the corresponding synapse according to Eq.(1).

$$net_j = \sum_{i=1}^{m} w_{ij} x_i + \theta_j \tag{1}$$

Where x_i is the ith component of the input signal, m is the dimension of the input signal, w_{ij} is the ith component of the synaptic weight value of neuron j, θ_j is the threshold value of neuron j, and net_j is the output value of the linear combiner.

3) The activation function, which limits the amplitude of the neuron output signal and limits the value to a certain value within the allowable range. In general, the normal amplitude of a neuron output signal value is in the range of [0, 1] or [-1, 1].

The output signal value y_j of the neuron model is calculated according to Eq.(2).

$$y_j = \varphi(net_j) \tag{2}$$

3.2. Multilayer Feedforward Neural Networks

In layer neural networks, neurons are organized by the layers. In the most simple layer neural networks, the source node constitutes the input layer. It is directly projected onto

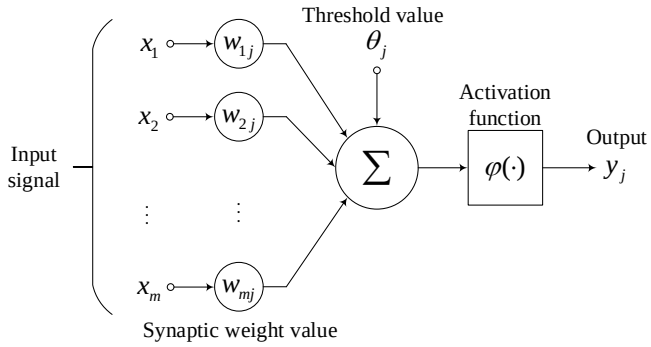

Fig. 1. Neuron model

the neuron of the output layer, rather than the opposite, which is called the feedforward neural networks. A multilayer feedforward neural network refers to one or more layers of hidden neurons in the neural network.

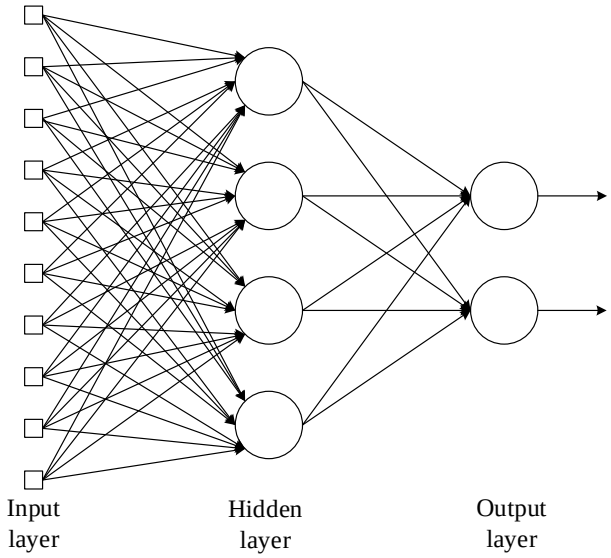

Fig. 2. Three-layer feedforward neural networks

The source nodes of the input layer provide an input signal to the hidden layer neurons. The input signal of each layer is the output signal of the previous layer. The output signal of one layer is used as the input signal of the next layer, and it is passed on. Finally, the output layer gives the neural network output signal of the corresponding original input signal. The structure is shown in Fig. 2. This is a 10-4-2 neural network with 10 source

nodes, 4 hidden neurons, and 2 output neurons. In general, a feedforward neural network has m source nodes, h neurons in the hidden layer and q neurons in the output layer, which can be called m-h-q neural networks. The output signal of output layer kth neuron is calculated according to Eq.(3).

$$z_k = \varphi_2 \left(\sum_{j=1}^{h} w_{jk} \varphi_1 \left(\sum_{i=1}^{m} w_{ij} x_i + \theta_j \right) + \theta_k \right) \tag{3}$$

Where $\varphi_1(\cdot)$ is the activation function of the hidden layer neuron, and $\varphi_2(\cdot)$ is the activation function of the output layer neuron.

3.3. Backpropagation Algorithm

Backpropagation (BP) algorithm is a common method to train artificial neural networks based on the gradient descent method. The gradient descent method minimizes the loss function by updating weight values and threshold values toward the fastest direction. Here, we derive the increment of weight values and threshold values.

For convenience of description, we use the square error function as the loss function according to the Eq.(4).

$$J(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{q} (t_k - z_k)^2 = \frac{1}{2} ||\mathbf{t} - \mathbf{z}||^2 \tag{4}$$

Where $\mathbf{t}$ and $\mathbf{z}$ are the target vector and the neural networks output vector whose length is q,$\mathbf{w}$ represents all the weight values in the neural networks.

The backpropagation rule is based on the gradient descent method. In the beginning, the weight values are initialized to random values and then adjusted toward the direction of reducing the error value.

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}} \tag{5}$$

This can be expressed in the form of a component,

$$\Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}} \tag{6}$$

Where η is the learning rate, which represents the relative change ratio of the weight values. We consider Eq.(6) in the three-layer neural networks.

1) We use the chain differentiation rule to entail the update rules of weight values and threshold values from the hidden layer to the output layer.

$$\frac{\partial J}{\partial w_{jk}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}} = -\delta_k \frac{\partial net_k}{\partial w_{jk}} \tag{7}$$

The sensitivity degree of unit k is defined as

$$\delta_k = -\frac{\partial J}{\partial net_k} \tag{8}$$

This sensitivity degree describes that the total error varies with the activation of the unit.

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k)\varphi_2{}'(net_k) \tag{9}$$

And due to

$$\frac{\partial net_k}{\partial w_{jk}} = y_j \tag{10}$$

So

$$\Delta w_{jk} = \eta\delta_k y_j = \eta(t_k - z_k)\varphi_2{}'(net_k)y_j \tag{11}$$

Same reason

$$\Delta\theta_k = \eta(t_k - z_k)\varphi_2{}'(net_k) \tag{12}$$

2) We entail the update rules of weight values and the threshold values from the input layer to the hidden layer.

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ij}} \tag{13}$$

Where

$$\begin{aligned}
&\tfrac{\partial J}{\partial y_j} = \tfrac{\partial}{\partial y_j}\left[\tfrac{1}{2}\sum_{k=1}^{q}(t_k - z_k)^2\right] \\
&= -\sum_{k=1}^{q}(t_k - z_k)\tfrac{\partial z_k}{\partial y_j} \\
&= -\sum_{k=1}^{q}(t_k - z_k)\tfrac{\partial z_k}{\partial net_k}\tfrac{\partial net_k}{\partial y_j} \\
&= -\sum_{k=1}^{q}(t_k - z_k)\varphi_2{}'(net_k)w_{jk}
\end{aligned} \tag{14}$$

So

$$\Delta w_{ij} = \eta\left[\sum_{k=1}^{q} w_{jk}\delta_k\right]\varphi_1{}'(net_j)x_i \tag{15}$$

Same reason

$$\Delta\theta_j = \eta\left[\sum_{k=1}^{q} w_{jk}\delta_k\right]\varphi_1{}'(net_j) \tag{16}$$

Where net_j is the weighted sum of the input x_j for the hidden layer neuron j, and net_k is the weighted sum of the input y_j for the output layer neuron k.

4. Artificial Bee Colony Algorithm

The artificial bee colony algorithm (ABC) is an algorithm that simulates honey bee collective behavior. Its role is divided into employed bee, onlooker bee, and scout bee. Assume that in the D dimensional space, the population size is $2 \times SN$, SN is the number of honey sources (employed bee number = onlooker bee number = SN). The honey sources and the employed bees correspond one to one, that is to say, the number of honey sources is SN, and the position of the $i\mathrm{th}$ source of honey is denoted as x_i. Each honey source location represents a candidate solution to the optimization problem, and the fitness of the honey source reflects the quality of the solution.

4.1. Four Phases of Artificial Bee Colony Algorithm

Initialization phase: At the beginning of the ABC, the initial food sources are generated randomly according to Eq.(17).

$$X_{i,j} = X_j^L + rand_j(X_j^U - X_j^L) \tag{17}$$

Where $i = \{1, 2, \ldots, SN\}$, $j = \{1, 2, \ldots, D\}$, X_j^L and X_j^U are the upper and lower bounds of the $j\mathrm{th}$ variable respectively. $rand_j$ is a random value in the range of [0,1]. Then, the fitness value of the food sources are calculated by Eq.(18).

$$fit_i = \begin{cases} \frac{1}{1+f(X_i)} & f(X_i) \geq 0 \\ 1 + |f(X_i)| & f(X_i) < 0 \end{cases} \tag{18}$$

Where fit_i is the fitness value of the food source X_i, $f(X_i)$ is the objective function value of food source X_i for the optimization problem. In addition, parameter $limit$ should be determined. The parameter $counter$ records the number of unsuccessful updates, is initialized to 0 for each food source.

Employed bee phase: Each Employed bee will fly to a distinct food source, looking for a candidate food source around the corresponding food source by using Eq.(19).

$$V_{i,j} = X_{i,j} + \phi_{i,j} \times (X_{i,j} - X_{k,j}) \tag{19}$$

Where i and k are randomly selected from $\{1, 2, \ldots, SN\}$, j is randomly selected from $\{1, 2, \ldots, D\}$, $V_{i,j}$ is the $j\mathrm{th}$ dimension of the $i\mathrm{th}$ candidate food source; $X_{k,j}$ is the $j\mathrm{th}$ dimension of the $k\mathrm{th}$ food source, $\phi_{i,j}$ is a random real number in the range of [-1,1].

After finding a new food source, the fitness value of the candidate food source is calculated by Eq.(18). If the candidate food source has a fitness value superior to the old food source, the food source replaces the old food source and the corresponding employed bee memorizes a new food source, and the $counter$ variable of this food source is reset to 0. Otherwise $counter$ is increased by 1.

Onlooker bee phase: According to the quality information of the food source shared by the employed bees, each onlooker bee will fly to a food source X_s, which is selected by the roulette wheel, in order to find a candidate food source according to Eq.(19). The probability of choosing the $i\mathrm{th}$ food source is calculated by Eq.(20).

$$p_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{20}$$

Obviously, the value of fitness is greater, the selection probability is higher. If a candidate food source V_s obtained by the onlooker bee is superior to the food source X_s, X_s will be replaced by the new food source. And the $counter$ variable of the food source is reset to 0. Otherwise $counter$ of the food source is increased by 1.

Scout bee phase: A food source with a highest $counter$ value is selected and compared with a predefined $limit$ value. If the value is greater than the value of $limit$, the selected food source is abandoned by its employed bee, and the employed bee is converted to a scout bee, and then this scout bee randomly finds a new source of food according to Eq.(17). After acquiring a new food source, the corresponding $counter$ is reset to 0 and the scout bee is changed to employed bee. Note that if the $j\mathrm{th}$ variable $V_{i,j}$ of the $i\mathrm{th}$ candidate food source violates the upper and lower bound constraints in the employed bee phase and the onlooker bee phase, the food source is reset according to Eq.(17).

4.2. The Improved ABC_elite Algorithm

ABC algorithm is a population-based, iteration-based and stochastic optimization algorithm, and it is very effective for solving numerical optimization problems. However, although the search equations are stronger in exploration, insufficient in exploitation. And the convergence performance of the ABC algorithm is not prominent. So, in order to better balance exploration and exploitation, Ref. [2] proposed a Depth First Search (DFS) framework and two search equations based on elite solutions, as shown in Eq.(21) and Eq.(22). This method named DFSABC_elite. The DFS framework can prioritize more computing resources to better solutions to enhance the exploitation ability of the algorithm. The elite-guided search equations keep the solution with the highest fitness value in the iteration, which speeds up the training process of the algorithm.

$$V_{i,j} = X_{e,j} + \phi_{i,j} \times (X_{e,j} - X_{k,j}) \tag{21}$$

$$V_{e,j} = \frac{1}{2}(X_{e,j} + X_{best,j}) + \phi_{e,j} \times (X_{best,j} - X_{k,j}) \tag{22}$$

Where X_e is a randomly selected solution from the elite solution, and X_k is a randomly selected solution from the current population. e is not equal to k, and k is not equal to i. X_{best} is the current best solution. $\phi_{i,j}$,$\phi_{e,j}$ are two random real numbers in the range of [-1, 1].

In order to better balance ABC exploration and exploitation capabilities, Ref. [6] addresses the problem that the candidate solution search equation has too large a disturbance to the elite solution in the search equation proposed in Ref. [2], and proposes an elite search equation. The candidate solutions for solutions and ordinary solutions should use different search equations. Ref. [17] proposed a novel search equation in Particle Swarm Optimization (PSO) algorithm, as shown in Eq.(23).

$$P_i = \frac{c_1 \times pbest_i + c_2 \times gbest}{c_1 + c_2} \tag{23}$$

Where c_1 and c_2 are two learning coefficients, $pbest$ is the personal best position, and $gbest$ is the population best solution found so far. Based on Eq.(23), a novel equation is proposed in Ref.[19].

$$X_i = N\left(\frac{gbest + pbest_i}{2}, |gbest - pbest_i|\right) \tag{24}$$

Where N represents the Gaussian distribution, $\frac{gbest+pbest_i}{2}$ represents the mean, and $|gbest - pbest_i|$ represents the standard deviation. The information around $pbest$ and $gbest$ is exploited by using the Gaussian distribution in Eq.(24). A similar Gaussian search equation is proposed according to Eq.(24).

$$V_{i,j} = N\left(\frac{X_{best,j} + X_{i,j}}{2}, |X_{best,j} - X_{i,j}|\right) \tag{25}$$

Where $X_{i,j}$ is the $j\mathrm{th}$ element of the elite solution X_i; $X_{best,j}$ is the $j\mathrm{th}$ element of the global best solution found so far, and j is randomly selected from $\{1, 2, ..., D\}$. According to Eq.(25), the elite solutions in employed bee phase search around X_{best} can improve the exploitation ability of ABC and the success rate of disturbance for elite solutions.

The search equation for the elite candidate solution is shown in Eq.(26).

$$V_{e,j} = \frac{1}{2}(X_{e,j} + X_{best,j}) + \phi_{e,j}(X_{best,j} - X_{e',j}) \tag{26}$$

The elite-guided search equations enhance ABC algorithm ability for exploiting solutions. The DFSABC_elite algorithm is combined with the two novel elite-guided search equations forming a new ABC variant called the Improved ABC_elite Algorithm (IABC_elite).

Define Fes (function evaluation) as the fitness function value, and $\max\_Fes$ (maximal function evaluation) as the maximal fitness function value. The pseudo code of the IABC_elite algorithm as follows:

Algorithm 1 The procedure of IABC_elite

Initialization: Generate SN solutions that contain variables according to Eq.(17);
while $Fes < \max\_Fes$ **do**
    Select the top solutions as elite solutions from populations;
    for $i = 1$ to SN **do**
        //employed bee phase
        if i is an elite solution **then**
            Generate a new candidate solution V_i in the neighborhood of X_i using Eq.(24);
        else
            Generate a new candidate solution V_i in the neighborhood of X_i using Eq.(21);
        end if
        Evaluate the new solution V_i;
        if $f(V_i) < f(X_i)$ **then**
            Replace X_i by V_i;
            counter(i)=0;
        else
            counter(i)=counter(i)+1;
        end if
    end for//end employed bee phase
    for $i = 1$ to SN **do**
        //onlooker bee phase
        Select a solution X_e from elite solutions randomly to search;
        $P_0 = 1 - Fes/\max\_Fes$
        if $rand(0,1) < P_0$ **then**
            Generate a new candidate solution V_e in neighborhood of X_e in neighborhood of X_e using Eq.(22)
        else
            Select a solution $X_{e'}$ from elite solutions randomly, where e' not equal to e;
            Generate a new candidate solution V_e using Eq.(26);
        end if
        Evaluate the new solution V_e;
        if $f(V_e) < f(X_e)$ **then**
            Replace X_e by V_e;
            counter(e)=0;
        else
            counter(e)=counter(e)+1;
        end if
    end for //end onlooker bee phase
    $Fes = Fes + SN * 2$
    Select the solution $X_{\max}$ with max counter value; //Scout bee phase
    if $counter(\max) > limit$ **then**
        Replace $X_{\max}$ by a new solution generated according to Eq.(17);
        $Fes = Fes + 1$,$counter(\max) = 0$
    end if//end scout bee phase
end while

5. BP Neural Networks Based on IABC_elite Algorithm

The Backprogation (BP) neural network based on IABC_elite algorithm is the combination of improved bee colony algorithm and BP neural networks algorithm. A new BP neural network optimized by improved ABC_elite algorithm is proposed and applied to intrusion detection. The research results of numerous papers show that the BP neural networks are sensitive to initial parameters [25,29,30]. Although there are many papers use particle swarm optimization (PSO), genetic algorithm (GA) and other algorithms to optimize the initial parameters of BP neural networks, the IABC_elite algorithm better balance the exploration and exploitation ability of the solution. With the strong ability to jump out of the local optimal solution, so using IABC_elite to optimize BP neural network can further improve the performance of BP neural networks. The optimized objection for IABC_elite algorithm are the weight values and threshold values of the neural networks. This method can improve the self-learning ability of BP neural network and speed up the convergence process of Backpropagation algorithm, so the neural network intrusion detection model have better detection ability. Its key techniques lie in:

Use the IABC_elite algorithm to train the neural network to generate the initial weight values and threshold values of the neural networks and use the error function of the neural networks as the fitness objective function of the IABC_elite algorithm.

The food source position in the IABC_elite algorithm represents the weight values and threshold values in the BP neural networks. The fitness value of each food source is calculated through the fitness function, which is the error function of the neural networks. The food source has a lower error value, has a higher fitness value. The IABC_elite algorithm tends to select the higher fitness value food source. The IABC_elite algorithm finds the honey source of the best fitness value in the search process. The best honey sources are obtained by comparing the current fitness value and historical value of each honey source within the process of iterations. The best honey source will be set as the initial weight values and threshold values of the BP neural network. When the error function value of the neural network generated by the best honey source meets the error precision requirement, the backpropagation algorithm performs further training for the neural network and tries to minimize the training error value. The algorithm avoids the neural network falling into a local optimum, achieves the goal of improving the accuracy and speeds up the neural network training. The basic idea of using IABC_elite algorithm to optimize BP neural network as follows:

First, we construct a neural network model, set the number of nodes in the input layer and the number of neurons in the hidden layer and the output layer. And construct the connection between the input layer and the hidden layer, as well as the hidden layer and the output layer. The activation function of the hidden layer neuron is set as Sigmoid function. The function equation is shown in Eq.(27), and the Sigmoid function curve is shown in Fig. 3. The Sigmoid function curve is S, which can map the output signal value of the linear combiner to the range of [0, 1].

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{27}$$

The activation function of the output layer neuron is set as Softmax function. This function is usually used in multi-classification process. It maps the output signal of multiple neurons to the range of [0, 1] and its equation is shown in Eq.(28).

$$Softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad j = 1, 2, \ldots, K \tag{28}$$

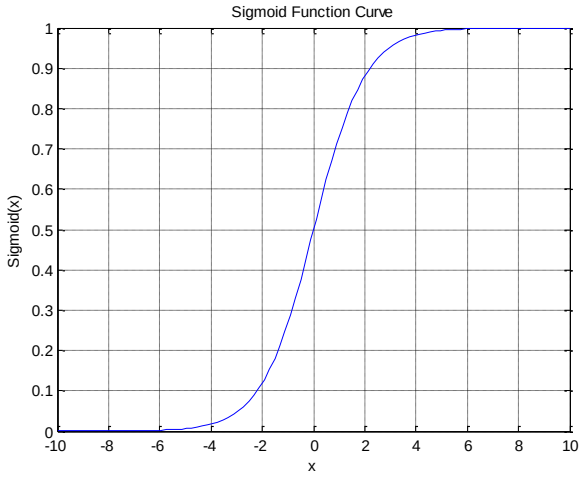

Fig. 3. Sigmoid Function Curve

Second, the IABC_elite algorithm is used to train the neural network and generate initial weight values and threshold values for the neural network. Set the population size of the IABC_elite algorithm and set the dimension of the solution according to the number of variables in the threshold values and weight values of the neural network. The position of the food source represents the weight values and threshold values of the neural network. The neural network error function is used as the food source fitness value function. High fitness food sources are those food sources with high fitness weight values and threshold values. The IABC_elite algorithm obtains a better solution by updating the honey sources during the iterative process. Table 1 show the mapping relation between neural networks training parameter and honey bee collective behavior. The IABC_elite algorithm choose the best fitness food source as the initial weight values and threshold values of the neural network.

Table 1. Mapping relation between neural networks training parameter and honey bee collective behavior

honey bee collective behavior	**neural networks training parameter**
Food source location	Weight values and threshold values
Food source fitness	Neural network error function value
High fitness food source	High fitness weight values and threshold values

Finally, the backpropagation algorithm trains the neural network according to the initial weight values and threshold values generated by IABC_elite algorithm. The backpropagation algorithm tries to minimize the training error by gradient descent. The weights and thresholds with the smallest training error will be used as parameters of the neural network for intrusion detection of network traffic.

The detail algorithm process is described as follows:

1) Select sample data for training, and randomly generate the weight values $\mathbf{w}_{ij}$ between the input layer neurons and the hidden layer neuron, the weight values $\mathbf{w}_{jk}$ between the hidden layer neurons and the output layer neurons. As well as generate the threshold values θ_j of the hidden layer neurons and the threshold values θ_k of the output layer neurons.

2) The output values of neural networks output layer neurons are calculated according to Eq.(29), Eq.(30), Eq.(31), and Eq.(32).

$$net_j = \sum_{i=1}^{m} w_{ij} x_i + \theta_j \tag{29}$$

$$y_j = \varphi_1(net_j) \tag{30}$$

$$net_k = \sum_{j=1}^{h} w_{jk} y_j + \theta_k \tag{31}$$

$$z_k = \varphi_2(net_k) \tag{32}$$

3) The error of the neural network is calculated according to Eq.(33). If it meets the required error, end training and go to step 5); otherwise go to step 4).

$$J(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^{q} (t_k - z_k)^2 \tag{33}$$

4) The weight values and threshold values between the hidden layer and the output layer are adjusted according to Eq.(34) and Eq.(35). The weight values and threshold values between the input layer and the hidden layer are adjusted according to Eq.(36) and Eq.(37).

$$\Delta w_{jk} = \eta (t_k - z_k) \varphi_2{}'(net_k) y_j \tag{34}$$

$$\Delta \theta_k = \eta (t_k - z_k) \varphi_2{}'(net_k) \tag{35}$$

$$\Delta w_{ij} = \eta \left[\sum_{k=1}^{q} w_{jk} \delta_k \right] \varphi_1{}'(net_j) x_i \tag{36}$$

$$\Delta \theta_j = \eta \left[\sum_{k=1}^{q} w_{jk} \delta_k \right] \varphi_1{}'(net_j) \tag{37}$$

Where

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k)\varphi_2{}'(net_k) \tag{38}$$

5) From the results obtained in step 4), we can get the new weight values $\mathbf{w}_{ij}$ and the new threshold values θ_j between the input layer and the hidden layer, as well as the new weight values $\mathbf{w}_{jk}$ and the new threshold values θ_k between the hidden layer and the output layer.

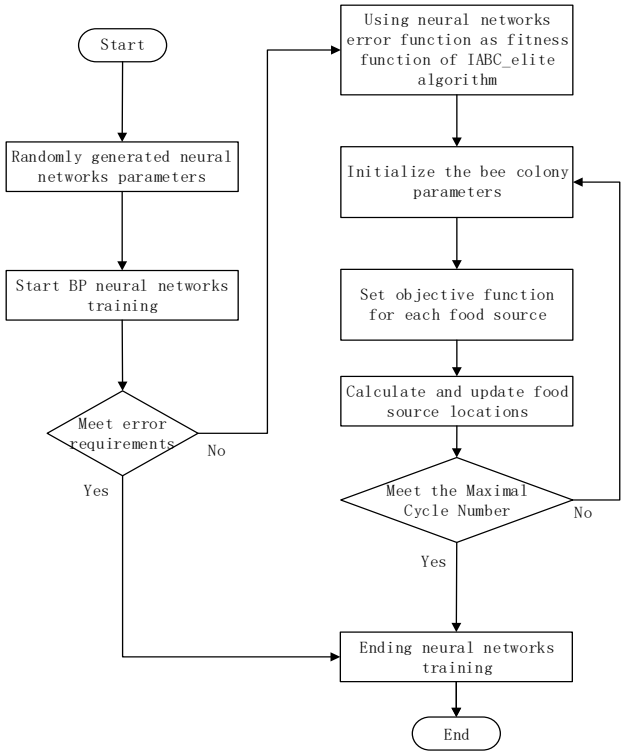

Fig. 4. Flowchart of IABC_elite algorithm to optimize BP neural networks

6) According to the new weight values and threshold values, re-execute step 1) to 3). If the error meets the requirements, end the training process. Otherwise, continue.

7) Using the current weights and thresholds as neural work input signal, and get the corresponding neural network output signal. The neural network loss function is set as the objective function of Eq. (39). Set the value of the Max Cycle Number (MCN).

$$fit_i = \begin{cases} \frac{1}{1+f(X_i)} & f(X_i) \geq 0 \\ 1 + |f(X_i)| & f(X_i) < 0 \end{cases} \tag{39}$$

8) Run the IABC_elite algorithm until iterations reaches the maximum number of iterations MCN. Set the weight values and threshold values from the IABC_elite algorithm as new initial parameters to train the neural network, then go to step 4).

9) End the training process; output the weight values $\mathbf{w}_{ij}$ and the threshold values θ_j as well as the weight values $\mathbf{w}_{jk}$ and the threshold values θ_k. Use the obtained neural networks model for testing data to predict the classification.

Table 2. Features of NSL-KDD datasets

No.	Features	Types	No.	Features	Types
1	duration	Continuous	22	is_guest_login	Symbolic
2	protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	srv_serror_rate	Continuous
6	dst_types	Continuous	27	rerror_rate	Continuous
7	land	Symbolic	28	srv_rerror_rate	Continuous
8	wrong_fragment	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	srv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_rate	Continuous
16	num_root	Continuous	37	dst_host_srv_diff_host_rate	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

6. Experiments and Analysis

6.1. Dataset Description

Prof. Sal Stolfo from Columbia University and Prof. Wenke Lee from North Carolina State University used data mining and other techniques to perform feature analysis and data preprocessing on DARPA 98 and DARPA 99 datasets, which form a new dataset. This dataset was used in the KDD CUP competition held in 1999 and is the benchmark dataset for intrusion detection. The dataset is named KDD CUP 99 dataset. But the KDD CUP 99 dataset has some shortcomings. For the shortcomings of the KDD CUP 99 dataset, the NSL-KDD data set removes the redundant data in the KDD CUP 99 dataset, overcoming the problem that the classifier is biased towards recurring records, and the performance of the learning method is affected [26]. In addition, the proper selection of normal

and abnormal network data ratios makes the testing and training data more reasonable, making it more suitable for efficient evaluation of different machine learning techniques. Table 2 shows the features of the NSL-KDD dataset. NSL-KDD dataset contains training dataset KDDTrain+.txt, KDDTrain+_20Percent.txt. KDDTrain+_20Percent is 20% of KDDTrain+ data. NSL-KDD dataset contains testing dataset KDDTest+.txt, KDDTest-21.txt. The test accuracy is generally less than KDDTest+ on KDDTest-21. The number of samples of different classifications of the NSL-KDD dataset shows in Table 3.

Table 3. Different Classifications of NSL-KDD Dataset

	KDDTrain+	**KDDTest-21**
Total	125973	11850
Normal	67343	2152
DoS	45927	4342
Probe	11656	2402
R2L	995	2754
U2R	52	200

6.2. Data Preprocessing

1) Vectorization

The NSL-KDD dataset has 34 numeric features and 7 symbol features. The features of 2, 3, 4, 7, 12, 21, and 22 columns are the symbol features, where the features of 7, 12, 21, and 22 columns are '0' or '1', and the features of 2, 3, and 4 columns are the character type. Because the input value of the neural networks should be a numeric matrix, we must convert some non-numeric features into numeric form.

Generally in machine learning, if there is an 'order' relationship between the feature values for symbol features, they can be converted to continuous values. For example, the value of the binary feature 'height' is 'high' and 'low', it can be converted to 1.0, 0.0. The value of the 3-value feature 'height' is 'high', 'middle' and 'low', it can be converted to 1.0, 0.5, 0.0; if there is no 'order' relationship between the feature values, the features values are usually converted into dimension vectors. For example, the value of the feature 'melon' is 'watermelon', 'pumpkin' and 'cucumber', it can be converted into (1,0,0), (0,1,0), (0, 0,1).

Thus, for example, the feature 'protocol_type' has three types of attributes, 'tcp','udp' and 'icmp'. It can be converted into (1, 0, 0), (0, 1, 0), (0, 0, 1).

2) Normalization

The normalization is to scale the data, which make it fall into a small specific range. In this paper, the data is uniformly mapped to the range of [-1, 1]. The normalization formula is shown in Eq. (40).

$$y = (ymax - ymin) * \frac{x - xmin}{xmax - xmin} + ymin \tag{40}$$

6.3. Evaluation Metrics

In the intrusion detection system, the most important performance measure is the accuracy of the detection, which means the classification ability of the neural network model. In order to evaluate the accuracy of the experiment, we introduce the concept of detection rate and false positive rate in the confusion matrix. The True Positive (TP) refers to the number of samples that the abnormal sample is identified as abnormal. The False Positive (FP) refers to the number of samples that the normal sample is identified as abnormal. The True Negative (TN) refers to the normal number of samples that the normal sample is identified as normal. The False Negative (FN) refers to the number of samples that the abnormal sample is identified as normal. Table 4 shows the definition of the confusion matrix. We have the following notation:

Accuracy (AC) is the percentage of the number of samples identified correctly over the total number of samples, as shown in Eq. (41).

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{41}$$

True Positive Rate (TPR) is the percentage of the number of anomaly samples correctly identified over the total number of anomaly samples, equal to the detection rate (DR), as shown in Eq.(42).

$$TPR = \frac{TP}{TP + FN} \tag{42}$$

False Positive Rate (FPR) is the percentage of the number of normal samples erroneously identified as anomaly samples over the total number of normal samples, as shown in Eq.(43).

$$FPR = \frac{FP}{FP + TN} \tag{43}$$

Therefore, the goal of the intrusion detection system is to obtain a higher accuracy and detection rate with a lower false positive rate.

Table 4. Confusion matrix

Actual Class \ Predicted Class	**anomaly**	**normal**
anomaly	TP	FN
normal	FP	TN

6.4. Experiment Results and Discussion

The experimental software environment for this paper is MATLAB 2014a. The experiment is performed on ThinkServer RD550, which has a configuration of two Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz and 16.0GB memory. There are five types of network packets in the NSL-KDD dataset. Therefore, we conducted experiments in five

categories: Normal, DoS, R2L, U2R, and Probe. In this experiment, KDDTrain+ is selected as the training dataset in NSL-KDD, and KDDTest-21 is selected as the testing dataset.

In Section 6.2, we map 41-dimensional features to 122 dimensions, so the input layer of the neural network has 122 nodes. The output result is 5 categories, so the output layer of the neural network has 5 nodes. We set the number of the hidden layer nodes of the neural network as 80 nodes. At this point, the neural networks need 10245 parameters to be optimized, so the solution in IABC_elite has a dimension of 10245. We set the population size of the bee colony as 100, the upper bound of the solution as 0.1, the lower bound as -0.1, and the maximum number of iterations of the IABC_elite algorithm as 100.

We select KDDTrain+ as the training dataset of BP neural network, and randomly take 30% of the data from it as a validation to avoid the BP neural network falling into overfitting. The cross-entropy is used as the error cost function of the backpropagation algorithm. In information theory, the cross entropy between two probability distributions over the same underlying set of events measures the average of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an 'unnatural' probability distribution, rather than the 'true' distribution. The average number of bits required for an event is also commonly used in error learning in machine learning. In order to speed up the neural network training process, the neural network is trained using the scaled conjugate gradient method [20].

Fig. 5 shows the BPNN cross entropy error iteration curve for randomly generating neural network initial parameters for the BP neural network training. Fig. 6 shows the ABC_BPNN cross entropy error iteration curve for the neural network training using the ABC algorithm pre-training parameters as initial parameters of the neural network. Fig. 7 shows the IABC_elite_BPNN cross entropy error iteration curve for the BP neural network training using the IABC_elite algorithm pre-training parameters as initial parameters of the neural network. The experimental results show that the initial parameters generated by the pre-training of the IABC_elite algorithm have lower initial errors, and the number of training epochs is smaller, which saves the time of neural network training.

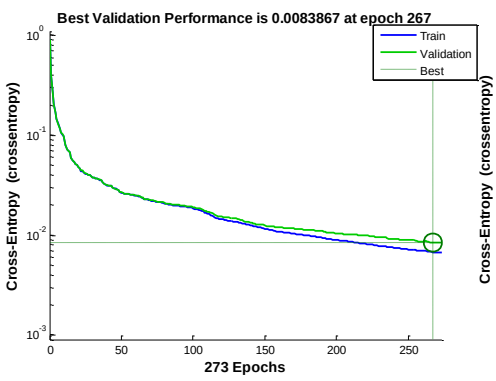

Fig. 5. BPNN Error Cost Iteration Curve

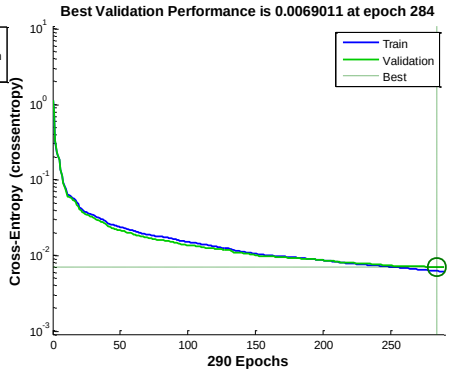

Fig. 6. ABC_BPNN Error Cost Iteration Curve

We test the neural network model on the KDDTest-21 testing dataset and use the five-classification confusion matrix to present the classification results. The class of labeling 1 in confusion matrix figure indicates the Normal class, the class of labeling 2 indicates the DoS class and the class of labeling 3 indicates the Probe class, the class of labeling 4 indicates the R2L class, and the class of labeling 5 indicates the U2R class. Fig. 8 is the classification result of five-classification confusion matrix of the neural network trained by a single BP algorithm. Fig. 9 is the classification result of five-classification confusion matrix of the neural network trained by BP algorithm optimized by ABC algorithm. Fig. 10 is the classification result of five-classification confusion matrix of the neural network trained by BP algorithm optimized by IABC_elite algorithm. Overall, the total prediction accuracy in Fig. 9 is 1.4% higher than Fig. 8. The total prediction accuracy in Fig. 10 is 2.8% higher than Fig. 9. In the view of the Normal class, the classification accuracy predicted by the BP neural network optimized by IABC_elite algorithm has been greatly improved, compared with the previous two figures. In the view of the five-classification confusion matrix, the BP neural networks optimized by IABC_elite algorithm have better generalization ability.

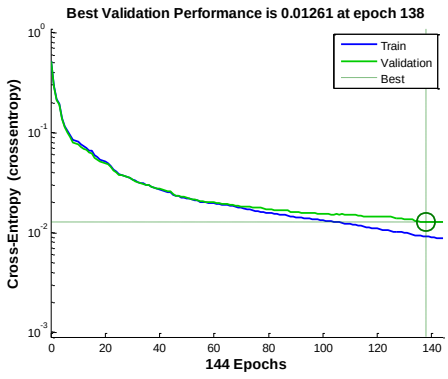

Fig. 7. IABC_elite_BPNN Error Cost Iteration Curve

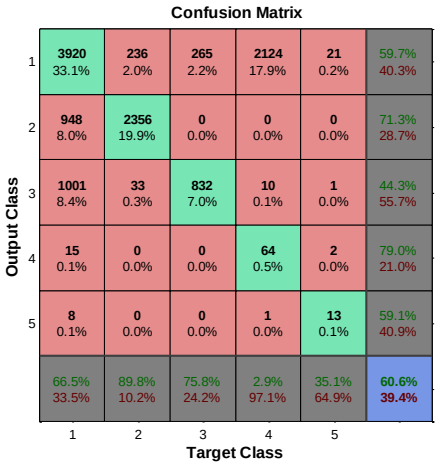

Fig. 8. BPNN Five-Classification Confusion Matrix

Therefore, in the training phase of the neural network, the BP neural network optimized by IABC_elite reduces the initial error of neural network training, requires less epoch training times, and save time of the neural network training. In the testing phase of the neural network, the BP neural network optimized by IABC_elite has better generalization ability, higher test accuracy and lower false positive rate, and better classification ability for normal and abnormal samples. This experiment results show that the initial parameter of the neural network generated by IABC_elite is better than the initial parameter generated by ABC and the randomly generated initial parameters. The initial weight values and threshold values generated by IABC_elite make it convenient to search

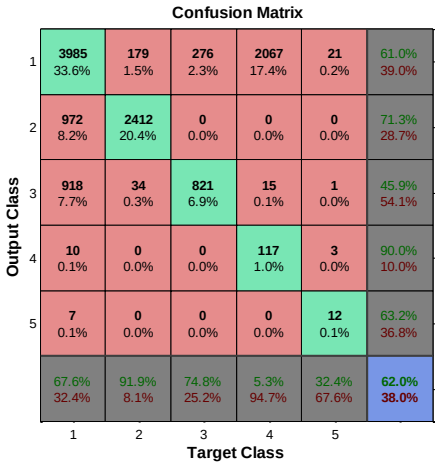

Fig. 9. ABC_BPNN Five-Classification Confusion Matrix

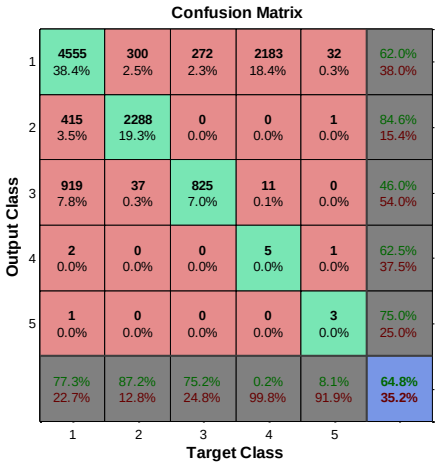

Fig. 10. IABC_elite_BPNN Five-Classification Confusion Matrix

the optimum weight values and threshold values for Backpropagation algorithm. It make the Backpropagation algorithm find a better solution which have a better generalization performance in testing phase of the neural network. The IABC_elite improved the convergence speed of the BP neural networks training and reduce the restrictions on initial weight values and threshold values, and avoid the neural network falling into a local optimum. The simulation verify the effectiveness of intrusion detection systems based on IABC_elite and BP neural networks.

7. Conclusion

This paper proposes a BP neural network model optimized by improved ABC_elite algorithm. The IABC_elite algorithm based on the depth first search framework better balances the ability of the exploration and exploitation of the artificial bee colony algorithm. It introduces the concept of elite solutions, uses two novel elite-guided search equations, and keep the highest fitness solutions. The highest fitness solutions accelerates the search process for optimal solutions. On the one hand, the IABC_elite improves the convergence speed of the BP neural network training; on the other hand, it reduces the restrictions to initial weight values and threshold values on the BP neural network training, and improves the robustness of the algorithm. We apply the neural network to intrusion detection, perform an experiment with NSL-KDD dataset, and create corresponding neural networks model according to the features of the NSL-KDD dataset. We select KDDTrain+ as training dataset from the NSL-KDD dataset to train the neural network, select KDDTest-21 as testing dataset to test the neural network. The confusion matrix is used to present the classification results, which shows that the neural network has better classification ability and verifies the effectiveness of the algorithm. The intrusion detection method IABC_elite and BP neural networks has a high detection rate and low false positive rate.

Acknowledgments. This work has been supported by the National Natural Science Foundation of China (No. 61672338 and No. 61873160).

References

1. Bullinaria, J.A., AlYahya, K.: Artificial bee colony training of neural networks. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2013), pp. 191–201. Springer (2014)
2. Cui, L., Li, G., Lin, Q., Du, Z., Gao, W., Chen, J., Lu, N.: A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. Information Sciences 367, 1012–1044 (2016)
3. Cui, M., Han, D., Wang, J.: An efficient and safe road condition monitoring authentication scheme based on fog computing. IEEE Internet of Things Journal pp. 1–1 (2019)
4. Deng, Y.: What is the future of disk drives, death or rebirth? ACM Computing Surveys (CSUR) 43(3), 23 (2011)
5. Denning, D.E.: An intrusion-detection model. IEEE Transactions on software engineering 13(2), 222–232 (1987)
6. Du, Z., Liu, G., Bi, K., Jia, J., Han, D.: An improved artificial bee colony algorithm with elite-guided search equations. Computer Science & Information Systems 14(3) (2017)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. John Wiley & Sons (2012)
8. Han, D., Bi, K., B., X., L., H., Wang, R.: An anomaly detection on the application-layer-based qos in the cloud storage system. Computer Science & Information Systems 13(2) (2016)
9. Han, D., Bi, K., Liu, H., Jia, J.: A ddos attack detection system based on spark framework. Computer Science & Information Systems 14(3) (2017)
10. Jaggi, R., Sangade, J.: Detecting and classifying attacks using artificial neural network. Int. J. Recent Innov. Trends Comput. Commun 2(5), 1136–1142 (2014)
11. Jiang, J., Wang, Z., Chen, T., Zhu, C., Chen, B.: Adaptive ap clustering algorithm and its application on intrusion detection. Journal on Communications 36, 118–126 (2015)
12. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, Computer Engineering Department (2005)
13. Karaboga, D., Akay, B., Ozturk, C.: Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In: Modeling Decisions for Artificial Intelligence. pp. 318–329. Springer Berlin Heidelberg (2007)
14. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. Journal of Global Optimization 39(3), 459–471 (Nov 2007)
15. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (abc) algorithm. Applied Soft Computing 8(1), 687–697 (2008)
16. Karaboga, D., Ozturk, C.: Neural networks training by artificial bee colony algorithm on pattern classification. Neural Network World 19(3), 279 (2009)
17. Kennedy, J.: Bare bones particle swarms. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706). pp. 80–87. IEEE (2003)
18. Lee, W., Stolfo, S.J., Mok, K.W.: A data mining framework for building intrusion detection models. In: Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344). pp. 120–132. IEEE (1999)
19. Lim, W.H., Isa, N.A.M.: An adaptive two-layer particle swarm optimization with elitist learning strategy. Information Sciences 273, 49–72 (2014)
20. Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural networks 6(4), 525–533 (1993)

21. Ozturk, C., Karaboga, D.: Hybrid artificial bee colony algorithm for neural network training. In: 2011 IEEE congress of evolutionary computation (CEC). pp. 84–88. IEEE (2011)
22. Qian, Q., Cai, J., Zhang, R.: Intrusion detection based on neural networks and artificial bee colony algorithm. In: 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS). pp. 257–262. IEEE (2014)
23. Qiu, C., Shan, J., Shandong, B., et al.: Research on intrusion detection algorithm based on bp neural network. International Journal of Security and its Applications 9(4), 247–258 (2015)
24. Revathi, S., Malathi, A.: A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research & Technology (IJERT) 2(12), 1848–1853 (2013)
25. SHEN Xiajiong, WANG Long, H.D.: Application of bp neural network optimized by artificial bee colony in intrusion detection. Computer Engineering 42(2), 190 (2016)
26. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6. IEEE (2009)
27. Wang, H., Wu, Z., Rahnamayan, S., Sun, H., Liu, Y., Pan, J.s.: Multi-strategy ensemble artificial bee colony algorithm. Information Sciences 279, 587–603 (2014)
28. Xie, J., Deng, Y., Min, G., Zhou, Y.: An incrementally scalable and cost-efficient interconnection structure for data centers. IEEE Transactions on Parallel and Distributed Systems 28(6), 1578–1592 (2017)
29. Yao, X.: Evolutionary artificial neural networks. International journal of neural systems 4(03), 203–222 (1993)
30. Yao, X., Islam, M.M.: Evolving artificial neural network ensembles. IEEE Computational Intelligence Magazine 3(1), 31–42 (2008)
31. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5, 21954–21961 (2017)
32. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied mathematics and computation 217(7), 3166–3173 (2010)

Letian Duan is currently a master student of Shanghai Maritime University. His research interests include cloud computing and Big Data.

Dezhi Han received the Ph.D. degree from the Huazhong University of Science and Technology. He is currently a Professor of computer science and engineering with Shanghai Maritime University. His research interests include cloud computing, mobile networking, wireless communication, and cloud security.

Qiuting Tian is currently a Ph.D. candidate at Shanghai Maritime University. Her research interests include cloud computing, mobile networking security and cloud security.

Received: October 1, 2018; Accepted: September 10, 2019.