Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 2511 – 2525
© TÜBİTAK
doi:10.3906/elk-1812-13

# A method for indoor Wi-Fi location based on improved back propagation neural network

**Jinghui CHEN**[1,3], **Chen DONG**[1,2,3*], **Guorong HE**[1,3], **Xiaoyu ZHANG**[1]
[1]College of Mathematics and Computer Science, Key Laboratory of Spatial Data Mining & Information Sharing, Ministry of Education, Fuzhou University, P.R. China
[2]Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou, P.R. China
[3]Fujian Provincial Key Laboratory of Information Security of Network Systems, Fuzhou, P.R. China

**Received:** 03.12.2018 • **Accepted/Published Online:** 15.04.2019 • **Final Version:** 26.07.2019

**Abstract:** In order to achieve high precision on indoor location, a Wi-Fi indoor location method based on improved back propagation (BP) neural network is proposed. The classical BP neural network is optimized in real time by the ant colony optimization algorithm. Meanwhile, the momentum term is introduced to construct an improved four-layer BP neural network model. The model uses the Wi-Fi signal feature as the input of the BP neural network and succeeds in the area classification under multiple Wi-Fi signal features. The experimental results demonstrate that the improved BP neural network can increase the classification accuracy of the classifier effectively, and achieve a high-precision indoor area location. Furthermore, it performs better practical results while ensuring the time complexity. The advantages of this method are high practicability, low cost, high prediction classification accuracy, and robust stability, which can achieve the efficient classification of the short-range area.

**Key words:** Indoor Wi-Fi location, back propagation neural network, ant colony optimization algorithm, momentum term

## 1. Introduction

With the rapid development of the wireless network, the accuracy of location is particularly concerned. The global positioning system (GPS) can satisfy people's location needs for outdoor environments well, but in relatively closed or complex situations, it often cannot obtain very accurate results and even correct area location, such as large buildings, shopping malls, and underground parking lots. Therefore, the indoor area location technology is given more attention.

Nowadays, there are many ways to achieve an indoor location, such as Bluetooth, radio frequency identification (RFID), and Wi-Fi. As an example of Bluetooth-based indoor location technologies, Lee et al. [1] used a Bluetooth receiver, an accelerometer, a magnetic field sensor, and a barometer on a smartphone to propose an indoor localization solution. Xinlong Jiang et al. [2] used Wi-Fi signal and Bluetooth low energy signal into a unified model to introduce a fusion semisupervised extreme learning machine. As an example of RFID-based indoor location technologies, He Xu et al. [3] used the weighted path length and support vector regression algorithm to solve the problem that the accuracy of the LANDMARC location algorithm relies on the density of reference tags and the performance of RFID readers. And He Xu et al. [4] also proposed a

*Correspondence: dongchen@fzu.edu.cn

novel indoor location algorithm based on Bayesian probability and k-nearest neighbor to reduce the location fluctuation and the error caused by multipath and environmental interference in LANDMARC. As an example of Wi-Fi–based indoor location technologies, SB Keser et al. [5] used multiple signal types (Wi-Fi–RSS and MF fingerprints) to propose an F-score-weighted indoor location algorithm to solve the problem that Wi-Fi signal distribution is nonstationary and accuracy is insufficient. Jingxue Bi et al. [6] considered the difference in signal strength caused by the orientation of users and proposed an indoor Wi-Fi location method based on an omnidirectional fingerprint database. Ban et al. [7] used pedestrian dead reckoning, Wi-Fi–based localization methods, and residual magnetism to propose a high-accuracy indoor location method.

Among them, the Wi-Fi signal feature location method has high location accuracy, low construction cost, and strong practicability. It is suitable for crowded areas and has broad application prospects. Machine learning is a method of using algorithms to parse data and then mine the underlying rules in the data to make decisions or predictions about certain things. In recent years, machine learning has been applied to the Wi-Fi–based positioning method and has achieved an excellent positioning effect [8–12]. Liao et al. [13] combined the decision tree classification model with indoor maps to reduce the dependence on hardware devices and improve the accuracy and reliability of the system. By normalizing the relation between K adjacent points in the k-nearest neighbor algorithm, Yeqing Fang et al. [14] improved the location accuracy and reduced the location matching time. Xiao et al. [15] used the BP neural network algorithm and the k-means algorithm for different scenarios to improve the detection rate in different situations and shorten the detection delay. However, there is still room for improvement in the positioning accuracy of these methods. Therefore, this paper aims to improve the accuracy of the Wi-Fi–based positioning method further.

This paper uses the BP neural network model's strong nonlinear mapping ability and good data fitting ability to adapt to noise data. Meanwhile, aiming at solving the problems of BP neural network's slow convergence speed and easy to fall into local minimum value, we use the ant colony optimization (ACO) algorithm to optimize the classical BP neural network in real time and introduce the momentum term to propose a Wi-Fi indoor location method based on improved BP neural network. Experiments show that this method can improve the accuracy of the indoor location method based on Wi-Fi signal features.

## 2. Preliminaries

### 2.1. Momentum term

To solve the problem that slow convergence speed and easy to fall into the local minimum value of the classical BP algorithm, the momentum term has been introduced to speed up the updating rate of the weight value and promote the accuracy rate of neural network prediction [16]. The weight calculation equation for input-hidden and hidden-hidden is as follows:

$$\omega_{ij}^{(l)} = \omega_{ij}^{(l)} - \eta_1 \cdot \frac{\partial E(\omega, b)}{\partial \omega_{ij}^{(l)}} + \gamma \cdot \Delta\omega_{ij}^{(l)} . \tag{1}$$

In Eq. (1), $\omega_{ij}^{(l)}$ represents the weight between the node $i$ of the layer $l$ and the node $j$ of the layer $l+1$, $\eta_1$ represents the rate of weight learning. $\frac{\partial E(\omega, b)}{\partial \omega_{ij}^{(l)}}$ represents the partial derivative of the error for the weight. $\gamma$ is the momentum term, and the value range is [0,1], it indicates the degree of influence caused by the weight changes of the previous reverse transfer subprocess in the current reverse transfer. $\Delta\omega_{ij}^{(l)}$ represents the weight used in the previous network calculation. Choosing the appropriate value of $\gamma$ can let the BP neural network to converge with a large learning rate. The initial parameters of the momentum term $\gamma$ can be determined by referring to the value of the learning rate [17]. When the learning rate is small, the momentum term $\gamma$ is often in the range [0.85,0.95]; when the learning rate is large, the momentum term $\gamma$ is often in the range [0.5,0.7].

$$E(\omega, b) = \frac{1}{2}\sum_{j=0}^{n-1}(d_j - y_j)^2 \,. \tag{2}$$

When performing forward and reverse transfer subprocesses for each training sample, Eq. (2) is used to determine whether the network error of a single sample meets the requirements. After iterating over the entire training sample, the mean error of the entire training sample is obtained using Eq. (3) to determine whether the network meets the training requirements:

$$\bar{E}(\omega, b) = \frac{1}{2}\sum_{j=0}^{n-1}(d_j - y_j)^2/sampleNum \,, \tag{3}$$

where $\omega$ is the weight, $b$ is the threshold, $d_j$ is the actual output of the network, $y_j$ represents the expected output of the sample, and $sampleNum$ indicates the total number of samples.

### 2.2. Hidden layer

In the process of neural network construction, increasing the number of hidden layers can reduce network errors and improve accuracy. However, it also complicates the network, which increases training time and overfitting. Therefore, we decided to use two layers of hidden layers. There are many methods to determine the number of hidden layer nodes, such as trial and error, pruning, growth, and direct typing methods. [18] The determination of the number of hidden layer nodes not only has a significant impact on the performance of the constructed BP neural network model but also is the direct cause of the overfitting phenomenon. This paper uses the direct typing method to determine the number of hidden layer nodes. The calculation equation is as follows:

$$hide = \sqrt{in + out} + \alpha \,, \tag{4}$$

where $in$ indicates the number of nodes in the input layer, $out$ shows the number of nodes in the output layer, $hide$ is the number of nodes in the hidden layer, and $\alpha$ is a constant in the range [1,10].

### 2.3. ACO–BP algorithm

Because of the simple principle and easy implementation, the BP neural network is widely used. However, there are also many problems in itself, such as slow convergence, sensitivity to initial values, and easy to fall into a local optimum. Therefore, Hong et al. [19] used the good global search ability of the ant colony algorithm to optimize the weight of the neural network.

The basic flow of the ACO–BP algorithm is as follows: First, assume that there are $m$ weights $p_1$, $p_2$, ..., $p_m$ in the network with the range of ($W_{min}$, $W_{max}$). The range of values of the weights is evenly divided into $N$ equal parts to form a set $I_{pi}$. The ant starts from the first initial weight and randomly selects a node according to the probability value in the current initial weight optional value set as the initial weight used in the round. Save the selected node number of each layer that the ant searched for as the ant path. The initial weight number to be selected is incremented by one, and the next candidate value is randomly selected, and record the path. The probability equation for selecting a node for each layer is as follows:

$$P[\tau_j^k(I_{pj})] = \frac{\tau_j(I_{pj})}{\sum_{j=1}^{N} \tau_j(I_{pj})}\,, \tag{5}$$

where $k$ is the number of each ant in the ant population, $I_{pi}$ is a set of optional values for a specific $w$, $\tau_j(I_{pj})$ is a pheromone of some optional value of $w$. In the initial state, the ants choose the probability of each node of each layer to be equal and completely random.

When an ant walks from the starting point to the end point, it saves the numbered path *antPath* of the selected node. The set of initial weights corresponding to the numbered path is taken out and brought into the BP neural network. After multiple training iterations of the training set samples in the BP neural network, calculate and save the mean error of the training set samples. After completing an iterative training for the whole ant population, according to the mean error of the sample obtained from the initial weight corresponding to the path selected by each ant to update the pheromone value of the entire node graph. The update equation is as follows:

$$\tau_j(I_{pj})(t+n) = \rho \cdot \tau_j(I_{pj})(t) + \Delta\tau_j(I_{pj})\,, \tag{6}$$

where $I_{pi}$ is a set of optional values for a specific $w$, $t+n$ is the search iteration of the whole ant every time, $n$ is the number of iterations, $\rho$ is the pheromone trail persistence, which is a constant in the range [0,1], $\Delta\tau_j(I_{pj})$ is a pheromone increment generated based on a sorting strategy, and its calculation equation is as follows:

$$\Delta\tau_j(I_{pj}) = \begin{cases} \dfrac{DBQ}{ant[k] \cdot pathAccu} & \text{if the ant } k \text{ selects the element } P_j(I_{pj}), \\ 0 & \text{else.} \end{cases} \tag{7}$$

where $DBQ$ represents the total amount of pheromone in the initial state, $ant[k].pathAccu$ is the mean error of the sample obtained after the ants selected a set of weights into the BP neural network for 500 iterations.

Repeat the above steps several times according to the preset number of iterations. During this process, record the best ant path continuously, and approach the optimal path gradually. After completing the repetitions, a set of initial weights corresponding to the optimal path are brought into the BP neural network for a comprehensive training iteration again. After completing the final training, the trained BP neural network model is used to predict the test set and predict the results.

## 3. Optimization of BP neural network

### 3.1. Activation function

Usually, the traditional neural network uses the Sigmoid function as the activation function and its forward transfer subprocess calculation formula as shown in Eq. (8).

$$S_j = \sum_{i=0}^{n-1} \omega_{ij}^{(l)} x_i + b_j\,, \tag{8}$$

where $\omega_{ij}^{(l)}$ represents the weight between the node $i$ of the layer $l$ and the node $j$ of the layer $l+1$, the threshold of node $j$ is $b_j$, and the output value of each node is $x_i$.

However, in the application background of this paper, the input range of the seven input features is in the range [–100,0]. If they are brought into the BP neural network for calculation directly, in the forward transfer subprocess, because $S_j$ is negative and its numerical amplitude changes greatly, so most of the values of $S_j$ fall on the left side of the Sigmoid function, where the slope is small, and the output value is close to zero infinitely. As a result, the gap between the outputs of the first layer of the BP neural network is too small to extract and distinguish features effectively, so that the training of the BP neural network fails. Therefore, it is necessary to preprocess the data before training the network model. The authors in [20] proposed an improved activation function model, as shown in Eq. (9).

$$f(x,a,b,c,d) = \frac{d}{(1+e^{[-a(x-b)]})} + c, \tag{9}$$

where $a$ adjusts the slope of the function, $b$ adjusts the left and right position of the function, $c$ adjusts the upper and lower position of the function. Moreover, $d$ adjusts the mapping interval range of the function. To make the Sigmoid function have stronger nonlinear mapping ability, modify the activation function as Eq. (10).

$$Sigmoid(x) = f(x) = \frac{A}{(1+e^{(\frac{-x}{B})})}, \tag{10}$$

where $A$ is an adjustment parameter in the vertical direction for appropriately adjusting the range of the values of Sigmoid function, and $B$ is an adjustment parameter in the horizontal direction for appropriately adjusting the tightness of the Sigmoid function.

## 3.2. Determination of BP neural network structure

There are seven input layer nodes to receive the different Wi-Fi signal strength values, because the sample data of the application background has seven input features. There are four output layer nodes, and each node output value indicates the probability that the positioning result is in the room. According to Section 2.2, we bring the number of hidden layer nodes corresponding to $\alpha$ of different values into the BP neural network and perform 4000 iterations of training. Table 1 shows the experimental results.

**Table 1**. The effect of $\alpha$ on the mean error.

| $\alpha$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mean error | 0.1807 | 0.1220 | 0.1472 | 0.1185 | 0.1442 | 0.1339 | 0.0657 | 0.0929 | 0.1074 | 0.1048 |

As shown in Table 1, when $\alpha$ is 7, the mean error of the sample is 0.0657, which is the smallest. According to Eq. (4), the number of hidden layer nodes of the BP neural network is 10. As a result, the BP neural network adopts a 7-10-10-4 four-layer structure which is shown in Figure 1. According to the BP neural network structure, there are 210 weights to be optimized, and the complexity is enormous. This paper optimized the 40 weight parameters between the third layer and the output layer to improve the efficiency of the algorithm properly. The range of each weight $w$ is [-1,1], and the range of each weight $w$ is divided evenly into ten values.

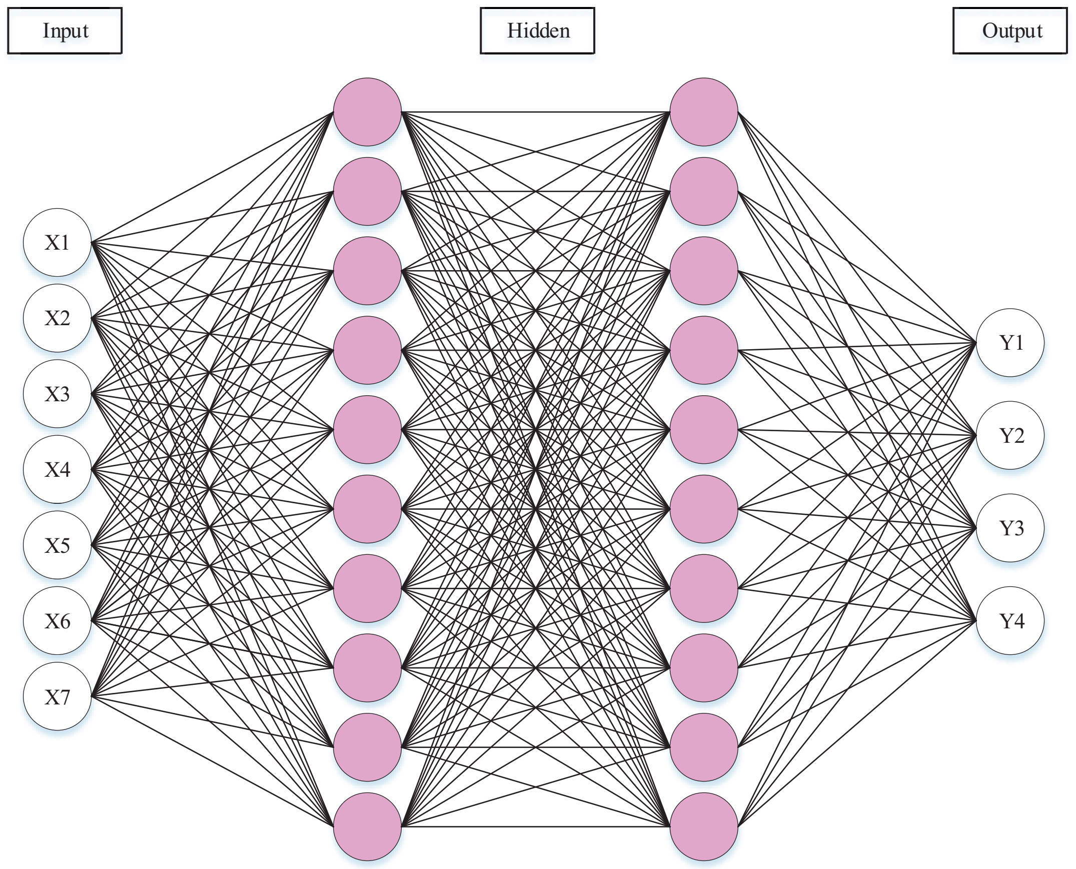

**Figure 1**. Structure of the BP neural network.

### 3.3. ACO–$\gamma$–BP algorithm

Initialize the parameters of the neural network before training the BP neural network. This paper further optimizes the BP neural network weight and threshold parameter initialization based on the Nguyen–Widrow algorithm [21]. Set the threshold $b$ of the two hidden layers and the output layer to 0. The weight $\omega$ of the hidden layer to the output layer is taken as a random number in the range [-1,1]. The weights and thresholds of the other layers are still generated using the Nguyen–Widrow algorithm. As a result, the values of the generated weights are all different. The iterative consistency can be avoided, and the returned gradient values will be moderate.

Due to the existence of random factors, BP neural network cannot fully ensure the optimal result of a minimum error in each training. Meanwhile, the BP neural network training model cannot achieve a higher accuracy prediction. Therefore, this paper proposes the ACO-$\gamma$-BP algorithm by combining the momentum term with the ACO–BP algorithm. Figure 2 shows a specific process. The steps are as follows:

Step 1. Initialize the parameters of the ACO algorithm, including the size of the ant colony, the number of iterations, and the optional set of initial weights;

Step 2. According to the probability calculation Eq. (5), one node is selected from the optional set of initial weights as the initial weight randomly;

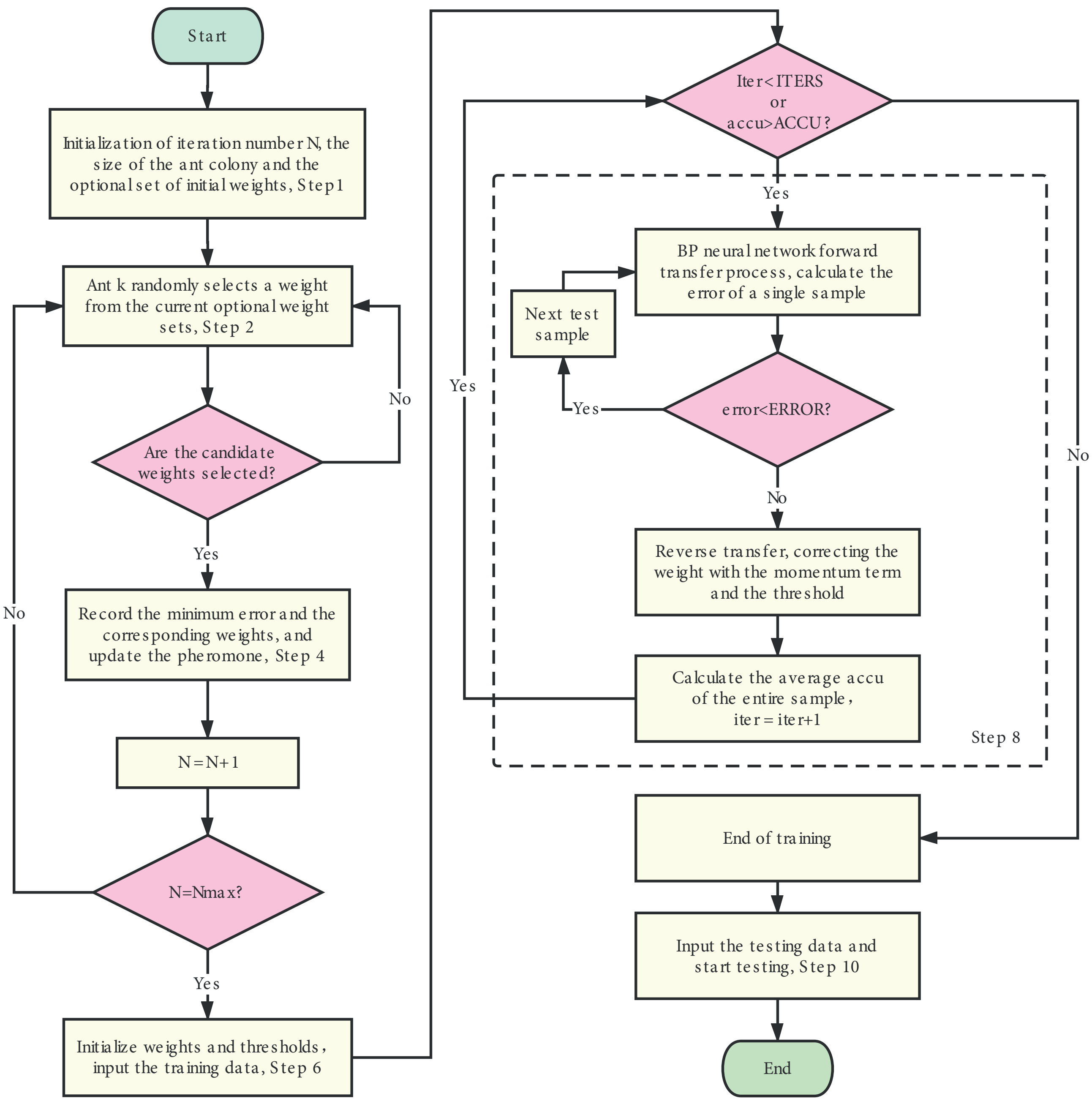
Start
Initialization of iteration number N, the size of the ant colony and the optional set of initial weights, Step1
Ant k randomly selects a weight from the current optional weight sets, Step 2
Are the candidate weights selected?
No
Yes
Record the minimum error and the corresponding weights, and update the pheromone, Step 4
N=N+1
N=Nmax?
No
Yes
Initialize weights and thresholds, input the training data, Step 6
Iter<ITERS or accu>ACCU?
Yes
No
BP neural network forward transfer process, calculate the error of a single sample
Next test sample
error<ERROR?
Yes
No
Reverse transfer, correcting the weight with the momentum term and the threshold
Calculate the average accu of the entire sample, iter=iter+1
Yes
Step 8
End of training
Input the testing data and start testing, Step 10
End

**Figure 2**. The flow of ACO–$\gamma$–BP neural network.

Step 3. Determine whether the candidate weights are selected. If all the candidate weights are selected, go to Step 4. Otherwise, return to Step 2;

Step 4. Bring the candidate weights into the BP neural network and perform several iterations. Record the minimum error and the corresponding weights. Meanwhile, update the pheromone and increase the number of iterations by 1;

Step 5. Determine whether the number of iteration $N$ reaches the maximum number of iteration $Nmax$. If yes, go to Step 6. Otherwise, return to Step 2;

Step 6. Bring the optimal weight into the BP neural network which combines with the momentum term. Initialize other parameters at the same time and input the training data.

Step 7. Perform several iterations. If the mean error $accu$ of the entire sample after each iteration is less than the preset value $ACCU$, or reach the set number of iterations $ITERS$, the loop is jumped out, and terminate the iteration. Otherwise, return to Step 8;

Step 8. For each sample, complete the forward transfer subprocess first, and then calculate the output error of each sample. If the output error of a single sample is less than the preset value $ERROR$, skip the sample loop, and enter the next sample. If the output error of the single sample is higher than the preset value $ERROR$, perform an inverse. The weight $\omega$ and threshold $b$ between each layer of the BP neural network are updated and perform the forward transfer again to calculate the output error until jumping out of the loop.

Step 9. When the preset number of iterations is completed or the average sample output error $accu$ is less than the preset value $ACCU$, the BP neural network training is completed basically.

Step 10. Each test sample is input into the trained neural network for testing.

## 4. Experiments

### 4.1. Dataset processing

The dataset of this experiment comes from the machine learning database of the University of California Irvine. The wireless indoor localization dataset is a collection of seven different indoor wireless network signal strengths that can be observed using a smartphone. The final output variable is the positioning result judged by the signal strength features of the seven wireless networks. The dataset was provided to the UCI database by Rajen Bhatt on December 4, 2017, with a total of 2000 instances [22].

Each instance contains seven input features and one output feature. The input features represent the Wi-Fi signal strength of seven different areas. The output feature is a value in the set {1, 2, 3, 4} corresponding to the room in which the Wi-Fi source is located. In order to adapt to the structure of the BP neural network algorithm used in the experiment, transform the output into a vector form with four values. The value 1 indicates that the positioning result is in the room, and the value 0 represents that the positioning result is not in the room. For example, [1, 0, 0, 0] indicates that the positioning result is room 1. Next, split the dataset of the 2000 instances into two parts. Part of it is a training set containing 1600 pieces of data (400 in each of four categories) for training in neural networks. The other part is a test set containing 400 pieces of data (100 in each of four categories).

### 4.2. Experimental and data analysis

In the experiment, the $ITRES$ is 5000, the $ACCU$ is set to 0.085, and the $ERROR$ is set to 0.03. The weight learning rate $\eta_1$ is 0.1, and the threshold learning rate $\eta_2$ is 0.1. Since the learning rate setting in this paper is relatively small, the momentum term is set to 0.8. In the first subprocess of the forward transfer of the neural network model, the parameters $A$ and $B$ of the activation function are set to 1 and 13. In other processes, the $A$ and $B$ are set to 1. The size of the ant colony is $2^3$. The pheromone trail persistence is 0.6. The total pheromone is initialized to 40.

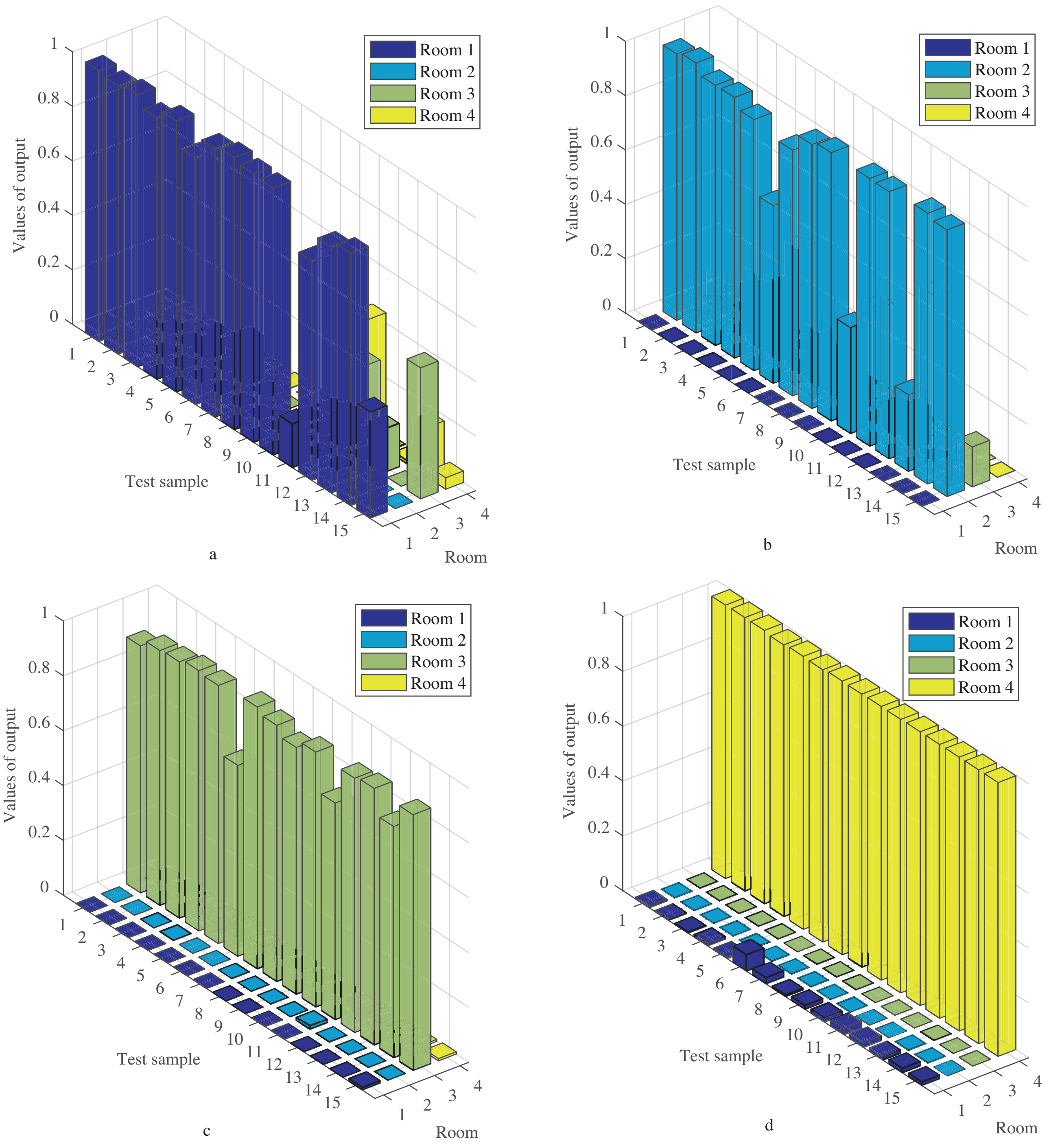

**Figure 3**. Prediction results for different rooms ((a) Room 1, (b) Room 2, (c) Room 3, (d) Room 4).

Table 2 and Figure 3 show the partial predictions after a single run of the improved BP neural network. Because the model uses the Sigmoid function, the output value of the model ($AO1$-$AO4$) does not indicate

**Table 2**. Single ACO–$\gamma$–BP simulation results.

| Test | PY1 | PY2 | PY3 | PY4 | AO1 | AO2 | AO3 | AO4 | T/F |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0.993268533 | 0.000000002 | 0.003485541 | 0.001493633 | T |
| 2 | 1 | 0 | 0 | 0 | 0.983780853 | 0.000000025 | 0.021965037 | 0.000607081 | T |
| 3 | 1 | 0 | 0 | 0 | 0.998478785 | 0.000000004 | 0.001485652 | 0.000326315 | T |
| 4 | 1 | 0 | 0 | 0 | 0.959101809 | 0.000000008 | 0.043312116 | 0.008095678 | T |
| 5 | 1 | 0 | 0 | 0 | 0.995024275 | 0.000000002 | 0.002216829 | 0.001174293 | T |
| 6 | 1 | 0 | 0 | 0 | 0.905691429 | 0.000000357 | 0.158920736 | 0.000606709 | T |
| 7 | 1 | 0 | 0 | 0 | 0.984765314 | 0.000000002 | 0.006848231 | 0.001803628 | T |
| 8 | 1 | 0 | 0 | 0 | 0.999140768 | 0.000000012 | 0.000936436 | 0.000022077 | T |
| 9 | 1 | 0 | 0 | 0 | 0.985861526 | 0.000000013 | 0.008530792 | 0.000149862 | T |
| 10 | 1 | 0 | 0 | 0 | 0.979401329 | 0.000000004 | 0.007739209 | 0.000717659 | T |
| 11 | 0 | 1 | 0 | 0 | 0.000000000 | 0.980556293 | 0.091323357 | 0.000000001 | T |
| 12 | 0 | 1 | 0 | 0 | 0.000000000 | 0.993481632 | 0.086216873 | 0.000000000 | T |
| 13 | 0 | 1 | 0 | 0 | 0.000000000 | 0.957889836 | 0.195528076 | 0.000000003 | T |
| 14 | 0 | 1 | 0 | 0 | 0.000000012 | 0.959448773 | 0.197769317 | 0.000000002 | T |
| 15 | 0 | 1 | 0 | 0 | 0.000000000 | 0.924403841 | 0.256712321 | 0.000000003 | T |
| 16 | 0 | 1 | 0 | 0 | 0.000000004 | 0.653875623 | 0.456297997 | 0.000000077 | T |
| 17 | 0 | 1 | 0 | 0 | 0.000000000 | 0.904311614 | 0.195353957 | 0.000000005 | T |
| 18 | 0 | 1 | 0 | 0 | 0.000000000 | 0.971412786 | 0.152003131 | 0.000000002 | T |
| 19 | 0 | 1 | 0 | 0 | 0.000000000 | 0.987799609 | 0.093390874 | 0.000000001 | T |
| 20 | 0 | 1 | 0 | 0 | 0.000000001 | 0.390585345 | 0.420841242 | 0.000000127 | F |
| 21 | 0 | 0 | 1 | 0 | 0.000028719 | 0.000018803 | 0.900775346 | 0.001764779 | T |
| 22 | 0 | 0 | 1 | 0 | 0.000007752 | 0.000001305 | 0.930297926 | 0.005612373 | T |
| 23 | 0 | 0 | 1 | 0 | 0.000200196 | 0.000004759 | 0.935257509 | 0.006583013 | T |
| 24 | 0 | 0 | 1 | 0 | 0.000000292 | 0.002012459 | 0.949207599 | 0.000029515 | T |
| 25 | 0 | 0 | 1 | 0 | 0.000010362 | 0.000000506 | 0.941713728 | 0.010483651 | T |
| 26 | 0 | 0 | 1 | 0 | 0.000000559 | 0.000002884 | 0.696347047 | 0.001081681 | T |
| 27 | 0 | 0 | 1 | 0 | 0.000310232 | 0.000001131 | 0.955610592 | 0.004590282 | T |
| 28 | 0 | 0 | 1 | 0 | 0.000200196 | 0.000004759 | 0.935257509 | 0.006583013 | T |
| 29 | 0 | 0 | 1 | 0 | 0.000028719 | 0.000018803 | 0.900778346 | 0.001764779 | T |
| 30 | 0 | 0 | 1 | 0 | 0.000007752 | 0.000001305 | 0.930297926 | 0.005612373 | T |
| 31 | 0 | 0 | 0 | 1 | 0.006738883 | 0.000000000 | 0.000013457 | 0.997514667 | T |
| 32 | 0 | 0 | 0 | 1 | 0.004402669 | 0.000000000 | 0.000008677 | 0.998492869 | T |
| 33 | 0 | 0 | 0 | 1 | 0.003962752 | 0.000000000 | 0.000001676 | 0.997923896 | T |
| 34 | 0 | 0 | 0 | 1 | 0.009824197 | 0.000000000 | 0.000107522 | 0.991364359 | T |
| 35 | 0 | 0 | 0 | 1 | 0.005869826 | 0.000000000 | 0.000049473 | 0.995958598 | T |
| 36 | 0 | 0 | 0 | 1 | 0.059235163 | 0.000000000 | 0.000033706 | 0.993421569 | T |
| 37 | 0 | 0 | 0 | 1 | 0.022794259 | 0.000000000 | 0.000005954 | 0.998547507 | T |
| 38 | 0 | 0 | 0 | 1 | 0.006709634 | 0.000000000 | 0.000023795 | 0.997248173 | T |
| 39 | 0 | 0 | 0 | 1 | 0.015133906 | 0.000000000 | 0.000000073 | 0.998500851 | T |
| 40 | 0 | 0 | 0 | 1 | 0.015162804 | 0.000000000 | 0.000008417 | 0.997600618 | T |

the probability of being located in a specific room. Take the room corresponding to the maximum value of the output value as the prediction result. In Table 2, $Test$ indicates the test sample number, $PY1 - PY4$ indicate the expected output values of the model; $AO1 - AO4$ indicate the actual output values of the model. Furthermore, $T$ represents the classification result true, and $F$ represents the classification result false. In Figure 3, the X-axis represents the number of the test sample, the Y-axis represents the predicted value of the model output, and the Z-axis represents the room number. The correct rate of this forecast is 89.75%. Of the 400 forecasted samples, 359 are accurate. The accuracy rates of the four types of areas are 92%, 87%, 83%, and 100%, respectively. This result also shows that the BP neural network in this paper is uniform in the prediction rate of each class reasonably.

**Table 3**. Multiple ACO–$\gamma$–BP simulation results.

| Time | Accuracy rate | Number of iterations |
|---|---|---|
| 1 | 92.50% | 2664 |
| 2 | 93.50% | 1910 |
| 3 | 93.75% | 890 |
| 4 | 92.50% | 1323 |
| 5 | 93.50% | 838 |
| 6 | 92.75% | 794 |
| 7 | 90.50% | 1961 |
| 8 | 88.75% | 3683 |
| 9 | 92.25% | 1522 |
| 10 | 90.50% | 1043 |
| 11 | 89.75% | 2181 |
| 12 | 90.50% | 4582 |
| 13 | 89.75% | 3649 |
| 14 | 90.25% | 2326 |
| 15 | 91.50% | 3009 |
| Average | 91.48% | 2158.33 |

To further analyze the correctness and usability of the proposed algorithm, we run the program multiple times and count the results of the experiments. Get and record 15 of the running experiments, as shown in Table 3. Considering the number of iterations when the algorithm reaches the preset $ACCU$, in the 15 cases of the above table, the minimum number of iterations can reach 800–1000 times. Because the weights and thresholds selected by the ACO algorithm are very robust, adapt most samples of the test set, in the iterative process. The convergence speed of the BP algorithm is very fast, so the number of iterations is small, and the prediction accuracy is also remarkably high.

Consider the case where the number of iterations is large, which needs to be more than 4000 times to reach the preset $ACCU$. By observing the mean error of the samples outputted in each iteration, after introducing the weight momentum term, the convergence of the algorithm in the early stage is stable and fast. However, when the mean error of the sample output reaches about 0.10 to 0.15, frequent fluctuations occur, and the back and forth oscillation causes the number of algorithm iterations to increase significantly. It is because during the BP neural network model training process, although the output mean error of the sample is similar, some of the samples have a large change in the weight of the entire BP neural network in the reverse transfer subprocess. This type of sample is called singular value which modifies the weights of the BP neural network during its own single sample training. Although the error of its own output is reduced, the changed weight causes the output error of other single training samples to increase greatly. Thus, the weight of the BP neural network has to be repeatedly adjusted and fluctuated back and forth. This situation cannot be entirely avoided. When the BP neural network trains the model, the specific value of the single sample output error falling to the preset value is unknown. Meanwhile, the change in the weight and the threshold is unknown. The appropriate increase in the preset iteration number can still reduce the output error of the BP neural network model to the preset value. Thus, the correctness can even be guaranteed.

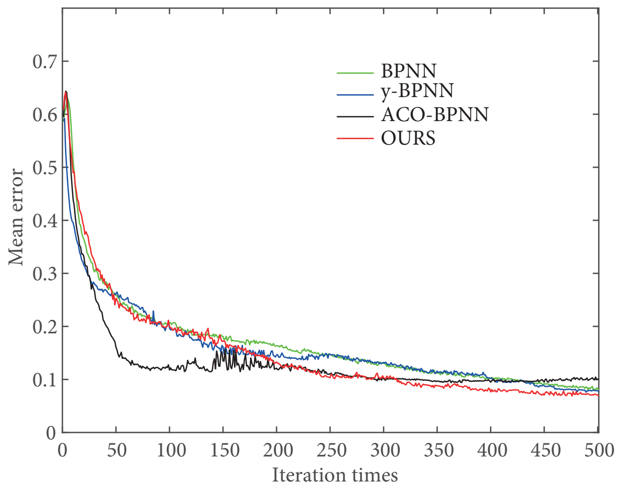

**Figure 4**. Comparison of training error.

The algorithm proposed in this paper mainly compares with the ACO–BPNN algorithm proposed by Li et al. [23]. In order to make the performance of this algorithm more persuasive, the algorithm also compares with the classical BP algorithm and the $\gamma$-BPNN algorithm which combines the BP algorithm and momentum term. In this paper, we perform several experiments on each of the four algorithms. Each algorithm is set to the same value on the same initialization parameters, and the other variables of the algorithm comparison experiment are unified. Table **??** shows the comparison results, while Figure 4 shows the downward trend of the mean error of each experimental training sample observed during the experiment. In Figure 4, the horizontal axis represents the number of iterations of the neural network, recorded every ten times, and the vertical axis represents the mean error of the entire training samples.

As can be seen from the statistical analysis results given in Table **??**, the proposed method can achieve the best average classification accuracy and the best average number of iterations. The algorithm proposed in this paper compares mainly with the ACO–BPNN algorithm. After introducing the momentum term, the ACO-BPNN algorithm has appeared multiple times to reach the preset error target value around 1000 iterations. However, the number of iterations of the ACO–BPNN algorithm without the momentum term is mostly between 2500 and 3500. After introducing the momentum term which the experimental results are shown in the last two columns of Table 4, the accuracy is increased by 0.17% only, but the average number of iterations is reduced by 20.46%. At the same time, combined with the experimental results in Figure 4, based on the ACO–BPNN algorithm, after introducing the momentum term, the fluctuation in the entire error decline process

**Table 4**. Statistics of multiple experimental results of four algorithms.

| Time | BPNN | | $\gamma$ -BPNN | | ACO-BPNN | | Ours | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Iteration | Accuracy | Iteration | Accuracy | Iteration | Accuracy | Iteration |
| 1 | 90.75% | 1086 | 89.25% | 1742 | 91.00% | 2161 | 92.50% | 2664 |
| 2 | 90.25% | 2410 | 90.50% | 2425 | 93.50% | 2659 | 93.50% | 1910 |
| 3 | 90.00% | 3800 | 92.00% | 2042 | 92.50% | 1642 | 93.75% | 890 |
| 4 | 91.75% | 1427 | 91.00% | 2741 | 89.75% | 3817 | 92.50% | 1323 |
| 5 | 90.75% | 2616 | 89.50% | 1305 | 91.50% | 3191 | 93.50% | 838 |
| 6 | 90.00% | 2731 | 88.00% | 2014 | 90.75% | 2357 | 92.75% | 794 |
| 7 | 93.75% | 1173 | 92.50% | 1610 | 92.50% | 3267 | 90.50% | 1961 |
| 8 | 92.25% | 2370 | 92.00% | 2128 | 89.00% | 2156 | 88.75% | 3683 |
| 9 | 90.75% | 4273 | 88.25% | 5000 | 90.50% | 2461 | 92.25% | 1522 |
| 10 | 92.50% | 3337 | 93.50% | 794 | 88.75% | 3683 | 90.50% | 1043 |
| 11 | 86.00% | 5000 | 90.75% | 3201 | 91.75% | 3181 | 89.75% | 2181 |
| 12 | 91.25% | 1900 | 92.25% | 3306 | 90.25% | 2326 | 90.50% | 4582 |
| 13 | 91.75% | 2032 | 91.50% | 2490 | 91.25% | 2117 | 89.75% | 3649 |
| 14 | 90.25% | 1016 | 92.25% | 1890 | 90.50% | 2924 | 90.25% | 2326 |
| 15 | 91.25% | 1945 | 90.25% | 2674 | 90.25% | 2762 | 91.50% | 3009 |
| Average | 90.88% | 2474.40 | 90.90% | 2357.40 | 91.32% | 2713.60 | 91.48% | 2158.33 |

is significantly reduced. Meanwhile, the network error decline speed is moderate, and the final error value significantly reduces.

## 5. Conclusions

Aiming at solving the problem that BP neural network is easy to fall into the local minimum value and slow convergence speed, which affects the accuracy of the Wi-Fi signal feature location method, this paper proposes a Wi-Fi indoor location method based on improved BP neural network. In this paper, we use the ACO algorithm to optimize the initial weight of the BP neural network. Meanwhile, to accelerate the update speed of the neural network weights, we also introduce the momentum term. Compared with other BP neural network models, the classification accuracy of the proposed algorithm is further improved and reduces the number of iterations effectively.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.61672159, No.61872091, No.U18042631, No.61702105), the Science Foundation of the Fujian Province, China (No.2018J01793, No.2018J01800) and the Foundation of the Education Department of Fujian Province, China (No.JAT170099).

## References

[1] Lee K, Nam Y, Min SD. An indoor localization solution using Bluetooth RSSI and multiple sensors on a smartphone. Multimedia Tools and Applications 2018; 77 (10): 12635-12654. doi: 10.1007/s11042-017-4908-2

[2] Jiang X, Chen Y, Liu J, Gu Y, Hu L. FSELM: fusion semi-supervised extreme learning machine for indoor localization with Wi-Fi and Bluetooth fingerprints. Soft Computing 2018; 22 (11): 3621-3635. doi: 10.1007/s00500-018-3171-4

[3] Xu H, Wu M, Li P, Zhu F, Wang R. An RFID indoor positioning algorithm based on support vector regression. Sensors 2018; 18 (5): 1504-1518. doi: 10.3390/s18051504

[4] Xu H, Ding Y, Li P, Wang R, Li Y. An RFID indoor positioning algorithm based on bayesian probability and k-nearest neighbor. Sensors 2017; 17 (8): 1806-1822. doi: 10.3390/s17081806

[5] Bozkurt KS, Yazici A, Gunal S. An f-score-weighted ındoor positioning algorithm ıntegrating wifi and magnetic field fingerprints. Mobile Information Systems 2018; 2018: 1-8. doi: 10.1155/2018/7950985

[6] Bi J, Wang Y, Cao H, Wang Y. A method of Wi-fi indoor positioning based on omnidirectional fingerprint database. Bulletin of Surveying and Mapping 2018; 2018 (2): 25-29. doi: 10.13474/j.cnki.11-2246.2018.0038

[7] Ban R, Kaji K, Hiroi K, Kawaguchi N. Indoor positioning method integrating pedestrian Dead Reckoning with magnetic field and WiFi fingerprints. In: IEEE 2015 Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU); Hakodate City, Hokkaido, Japan; 2015. pp. 167-172.

[8] Zhou R, Li Z, Luo L. WiFi-pedestrian dead reckoning fused indoor positioning based on particle filtering. Journal of Computer Applications 2016; 36 (5): 1188-1191,1200. doi: 10.11772/j.issn.1001-9081.2016.05.1188

[9] Shen X, Bai Z, Dong Y, Dong Y. WiFi indoor localization algorithm based on dynFWA-SVM. Transducer and Microsystem Technology 2018; 37 (4): 121-124,128. doi: 10.13873/J.1000-9787(2018)04-0121-04

[10] Li J, He X, Cai Y, Xu Q. Method of Wi-fi indoor location based on k—means and random forest. Engineering of China 2017; 24 (4): 787-792. doi: 10.14107/j.cnki.kzgc.15C6.0543

[11] Chang Q, Velde SVD, Wang W, Li Q, Hou H et al. Wi-Fi fingerprint positioning updated by pedestrian dead reckoning for mobile phone indoor localization. In: The 6th China Satellite Navigation Conference (CSNC 2015); Xian, China; 2015. pp. 729-739.

[12] Wang J, Zhang X, Gao Q, Yue H, Wang H. Device-free wireless localization and activity recognition: A deep learning approach. IEEE Transactions on Vehicular Technology 2017; 66 (7): 6258-6267. doi: 10.1109/TVT.2016.2635161

[13] Liao JK, Chiang KW, Tsai GJ, Chang HW. A low complexity map-aided fuzzy decision tree for pedestrian indoor/outdoor navigation using smartphone. In: IEEE 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN); Alcala de Henares, Spain; 2016. pp. 1-8.

[14] Fang Y, Deng Z, Xue C, Jiao J, Zeng H et al. Application of an improved k nearest neighbor algorithm in WiFi indoor positioning. In: The 6th China Satellite Navigation Conference (CSNC 2015); Xian, China; 2015. pp. 517-524.

[15] Xiao F, Guo Z, Zhu H, Xie X, Wang RC. AmpN: Real-time LOS/NLOS identification with WiFi. In: IEEE 2017 International Conference on Communications (ICC); Paris, France; 2017. pp. 1-7.

[16] Moallem P, Ayoughi SA. Improving backpropagation via an efficient combination of a saturation suppression method and momentum term. Neural Network World 2010; 20 (2): 207-222.

[17] Zhang N, Wu W, Zheng G. Convergence of gradient method with momentum for two-layer feedforward neural networks. IEEE Transactions on Neural Networks 2006; 17 (2): 522-525. doi: 10.1109/tnn.2005.863460

[18] Fan J, Wang Z, Qian F. Research progress structural design of hidden layer in BP artificial neural networks. Control Engineering of China 2005; 12 (5): 105-109. doi: 10.14107/j.cnki.kzgc.2005.s1.036

[19] Hong B, Jin F, Gao Q. Multi-layer feedforward neural network based on ant colony system. Journal of Harbin Institute of Technology 2003; 35 (7): 823-825. doi: 10.3321/j.issn:0367-6234.2003.07.017

[20] Li E, Yang P, Sun X. Improved algorithm of BP neural networks based on the Activation function with four adjustable parameters. Microelectronics & Computer 2008; 25 (11): 89-93. doi: 10.19304/j.cnki.issn1000-7180.2008.11.023

[21] Liu Q, Chen G, Liu X, Zhan J. Research on initialization algorithms of weights and biases of BP neural network. Journal of Southwest China Normal University 2010; 35 (6): 137-141. doi: 10.13718/j.cnki.xsxb.2010.06.050

[22] Rohra JG, Perumal B, Narayanan SJ, Thakur P, Bhatt RB. User localization in an ındoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks. In: Proceedings of Sixth International Conference on Soft Computing for Problem Solving, SocProS 2016; Thapar University, Patiala, India; 2016. pp. 286-295.

[23] Li X, Tong Z, Guo E, Luo X. Quantifying spatiotemporal dynamics of solar radiation over the northeast China based on ACO-BPNN Model and Intensity Analysis. Advances in Meteorology 2017; 2017: 1-15. doi: 10.1155/2017/1042603