Hindawi
Journal of Control Science and Engineering
Volume 2018, Article ID 6565737, 10 pages
https://doi.org/10.1155/2018/6565737

Research Article

Data Preprocessing Method and Fault Diagnosis Based on Evaluation Function of Information Contribution Degree

Siyu Ji and Chenglin Wen

School of Automation, Hangzhou Dianzi University, Hangzhou 310018, China

Correspondence should be addressed to Chenglin Wen; wencl@hdu.edu.cn

Received 18 February 2018; Revised 14 April 2018; Accepted 30 April 2018; Published 2 July 2018

Academic Editor: Youqing Wang

Neural network is a data-driven algorithm; the process established by the network model requires a large amount of training data, resulting in a significant amount of time spent in parameter training of the model. However, the system modal update occurs from time to time. Prediction using the original model parameters will cause the output of the model to deviate greatly from the true value. Traditional methods such as gradient descent and least squares methods are all centralized, making it difficult to adaptively update model parameters according to system changes. Firstly, in order to adaptively update the network parameters, this paper introduces the evaluation function and gives a new method to evaluate the parameters of the function. The new method without changing other parameters of the model updates some parameters in the model in real time to ensure the accuracy of the model. Then, based on the evaluation function, the Mean Impact Value (MIV) algorithm is used to calculate the weight of the feature, and the weighted data is brought into the established fault diagnosis model for fault diagnosis. Finally, the validity of this algorithm is verified by the example of UCI-Combined Cycle Power Plant (UCI-ccpp) simulation of standard data set.

1. Introduction

In this paper, based on the evaluation function to calculate the weight of the data feature, the premise is to evaluate the accuracy of the function, which is the theoretical basis and guarantee for the validity of the follow-up work. At present, BP neural network is one of the most popular regression algorithms. It can effectively approximate complex nonlinear mappings based on inputting samples and has the advantages of simple structure and high operability and has been applied in many fields [1, 2]. On the other hand, there are some problems [3]. First of all, the parameter training method based on the gradient descent learning algorithm converges slowly and falls into the local optimum. Secondly, there are many parameters in the neural network that need to be trained, which can take a great time to deal with. In the actual operation of the system, the model data of the system may not have been completely collected but acquired one by one or block by block, while the neural network is a learning algorithm lacking the ability to update online. It is unacceptable for many practical situations to retrain the network in response to changes in the modalities of system. However, if the parameters in the neural network are not updated in time, the fitting output of the network will greatly deviate from the real value.

In order to solve the above problems, the industry has proposed Neural Network Incremental Learning algorithm to deal with them [4]. The incremental learning method can adjust the artificial neural network by analyzing the specific conditions and recognition results of each new sample, learn new knowledge based on the existing knowledge, and flexibly adapt to the dynamic changes of the environment [5]. Therefore, there are many researches on incremental learning algorithms [6, 7]. The main idea of incremental learning is mainly reflected in two aspects: (1) in the actual perception of data, the amount of data is often gradually increased. Therefore, in the face of new data, the learning method should only update the changes caused by the new data without changing the original knowledge base and then learn from the new data contained in the knowledge; (2) the

cost of modifying a well-trained system is usually less than the cost of retraining one system. There are many frameworks for incremental learning. The core of each framework is to evaluate the similarity between new data and stored knowledge. The method thus determines the way in which new knowledge is perceived and the knowledge base is increased, which affects the growth of knowledge. Therefore, the judgment mechanism of new knowledge is the core part of incremental learning. In an Orthogonal Least Square (OLS) learning algorithm proposed by Chen, structure and parameter identification are performed simultaneously [8]. In [9], a dynamic fuzzy neural network (DFNN) based on RBF is proposed. The parameters are adjusted by Linear Least Square (LLS), and the structure can be adaptively added and deleted according to the rules. On the basis of these, a generalized DFNN (GDFNN) is proposed by the literature [10]. Based on the Ellipse Basis Function (EBF), an online parameter locating mechanism is proposed. The above algorithms are more difficult to achieve and at the same time cannot guarantee the real-time algorithm.

The Kalman filtering method was widely used in the fields of process control, communication, biomedical science, etc., once proposed in the 1960s. Because of its recursive nature, it does not need to store a large amount of historical information and reduces the amount of computer storage. Combining the system state equation and the observation equation directly, it can directly give the estimation accuracy when estimating system state parameters. Its concise way of thinking had become the theoretical basis for the development of such theories as estimation theory and emerging information fusion [11, 12]. Compared with Kalman, the least squares method uses all the observed data to estimate the value of the state quantity at the initial time. Due to the large number of observations and the statistical characteristics of the method, then this method has a higher accuracy. However, due to its centralized nature, it lacks real-time performance. Kalman filter after the observation data is updated, the state variables are improved with new observation data to obtain the state variables at this observation time. Therefore, Kalman filtering is suitable for real-time processing. In this paper, a method of real-time update of hidden layer output weights based on Kalman filter is proposed, which avoids the retraining of the model. At the same time, the deviation from the real value of the model output data caused by the failure of the model parameters to be updated due to the system modal change is eliminated.

The rest of this article is organized as follows: Section 2 gives a brief description of the problems to be solved. The formal description of BP neural network, the gradient descent method, least squares method, and Kalman filter method are used to update the global or local parameters of the network in Section 3. In Section 4, we give a detailed introduction to the process of weighting MIV algorithm. Section 5 simulates the proposed algorithm based on the UCI standard dataset and results of comparisons and analyses. Section 6 summarizes the related research contents, elaborates on the existing problems, and gives a prospect of the next work.

2. Problem Description

Based on the data-driven fault diagnosis method, a series of data processing is often required before the fault diagnosis, such as data standardization, data dimension reduction, feature selection, feature weighting, etc. Some even need to map the data onto high-dimensional space, like we are using support vector machine (SVM) for data classification. The ultimate goal of all the above data preprocessing operations is to improve the diagnostic performance of the fault diagnosis method. Different data preprocessing operations are often adopted for different fault diagnosis methods. Feature weighting algorithm is relatively special, and no matter which kind of data preprocessing operation, you can then weight their features. The purpose of the weight of the feature is to amplify the difference between the feature variables of the data and to eliminate the feature redundancy to a certain extent.

$X = [x(1), x(2), \cdots, x(N)]$ is the data set composed of N samples and $x(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^T$ is a sample of the data set X, where $x(k)$ is the sample at time k and $x_i(k)$ is the component at time k, $k = 1, 2, \cdots, N$; $i = 1, 2, \cdots, n$. Our purpose is to find a transformation matrix P to perform a weighted transformation on the sample data

$$\overline{x}(k) = Px(k) \tag{1}$$

$$\overline{x}(k) = \begin{bmatrix} p_1 & & & \\ & p_2 & & \\ & & \ddots & \\ & & & p_n \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} = \begin{bmatrix} p_1 x_1(k) \\ p_2 x_2(k) \\ \vdots \\ p_n x_n(k) \end{bmatrix} \tag{2}$$

Making the weighted transformed sample data $\overline{x}(k)$ can be more effective in fault diagnosis. The next question is how to model and find the transformation matrix P: the traditional principal component analysis (PCA) algorithm can solve the transformation matrix to map the original data containing n features onto the selected m-dimensional feature space and have $m < n$, so PCA lost some information in the process of data transformation. The information entropy feature weighting algorithm weights the sample data based on the uncertainty of the feature set's classification of the data set, but its calculation is more difficult; the data set also has higher requirements. In this paper, we want to obtain the weight of the feature through an evaluation function. The evaluation function is generated based on the BP neural network fitting function. In order to ensure the accuracy of the evaluation function, we generally need to retrain the network when the system modal changes. This is a time-consuming task. In order to avoid the retraining of the model, this paper proposes a Kalman filter-based algorithm for real-time update of hidden layer output weights of BP neural network and then establishes the evaluation function model to obtain the transformation matrix (i.e., feature weights). Following we are going to introduce the establishment of the evaluation function and the weight of the data feature in detail. The algorithm implementation process is shown in Figure 1.

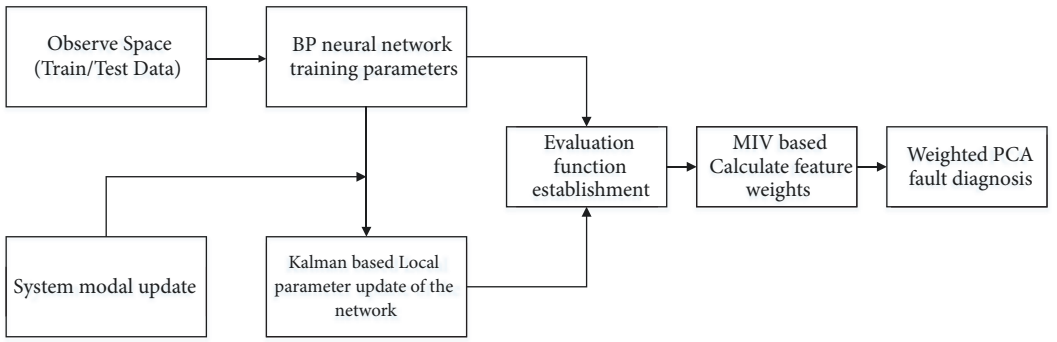


Figure 1: Evaluation function of information contribution degree algorithm process.

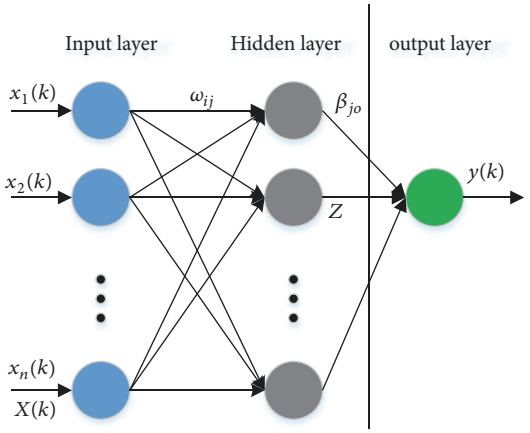


Figure 2: BP neural network structure.

3. Evaluation Function Established

The weight of the feature in the above problem depends on the evaluation function. The evaluation function model is generated based on the fitting function of the neural network (i.e., the algebraic representation of the model), and the influence of the input to the output is calculated by the Mean Impact Value (MIV) algorithm, so this article refers to it as the evaluation function. In order to ensure the accuracy of the evaluation function, a precise system model will be necessary. Next, we will conduct a modeling analysis on how to build a model that is accurate and the model parameters updated in real time as the system modal changes.

3.1. Formal Description of BP Neural Network. BP neural network is a kind of feedforward neural network; its main feature is the signal forward, the error of the reverse transmission. In the forward transfer process, the signal input is processed by the hidden layer and finally output. If the output layer error is greater than our set value, it is propagated backwards. After this is repeated, constantly adjust the network weights and bias until the output error meets the set value, or the number of iterations of the algorithm goes up to the ceiling [13]. BP neural network structure is shown in Figure 2.

For the dataset $X = [x(1), x(2), \cdots, x(N)]$, we need corresponding output $Y = [y(1), y(2), \cdots, y(N)]$ in neural network training and bring the above labeled dataset into the established BP neural network structure model as (3) to train the model parameters and get the parameters to estimate ω, a, β, b

$$y(k) = \sum_{j=1}^{l} \beta_{jo} f\left(\sum_{i=1}^{n} \omega_{ij} x_i(k) + a_j\right) + b, \quad k = 1, 2, \cdots, N \tag{3}$$

where ω and β are input layer output weight matrix and hidden layer output weight matrix, a and b are the hidden layer bias and output layer bias vector, $\omega \subseteq R^{l \times n}$, $\beta \subseteq R^{l \times m}$ and $a \subseteq R^{l \times 1}$, $b \subseteq R^{m \times 1}$, respectively. BP neural network can be regarded as a nonlinear function, neural network output value, and predicted value of the function of the independent variables and dependent variables. When the input vector is n-dimensional and the output vector is m-dimensional, the BP neural network expresses the function mapping from n-dimensional space onto m-dimensional space.

BP neural network conducts network training before making predictions; the training process includes the following steps.

Step 1. Initialization of neural network parameter: according to the input and output sequence of the system, the number of hidden layer nodes l and the number of output layer nodes and m are determined. The weights ω and β of the input layer to the hidden layer and the hidden layer to the output layer are initialized, and the bias vectors are a and b. Set the learning rate η and select the activation function.

Step 2. Calculating output of hidden layer: according to the input of input layer, input layer to the hidden layer of the weight, and bias, calculate output of hidden layer z

$$z_j(k) = f\left(\sum_{i=1}^{n} \omega_{ij} x_i(k) + a_j\right), \quad i = 1, 2, \cdots, n; \ j = 1, 2, \cdots, l \tag{4}$$

where f is the hidden layer of the excitation function and the function has many forms, common like sigmoid function, and Gaussian function.

Step 3. Calculating output of output layer z_{jk}: according to output of hidden layer β, output weight of hidden layer vector a, and bias vector b, we calculate the BP neural network prediction output value $\widehat{y}$

$$\widehat{y}(k) = \sum_{j=1}^{l} \beta_{jo} f\left(\sum_{i=1}^{n} \omega_{ij} x_i(k) + a_j\right) + b, \quad o = 1, 2, \cdots, m \tag{5}$$

Step 4. Error calculation: based on the predicted output $\widehat{y}(k)$ of system and expected output $y(k)$, a network prediction error e is calculated

$$e = \sum_{k=1}^{L} \left[\widehat{y}(k) - y(k)\right] \tag{6}$$

Step 5. Updating of weight vector and bias vector: we are going to describe in detail the different weight update algorithms in the following section.

Step 6. Determine whether $\|e(k)\|$ is smaller than the set value or whether the number of iterations of the algorithm has reached the ceiling. If it is "yes" the algorithm ends; if it is "no" the algorithm returns to Step 2.

3.2. Estimation of Evaluation Function Parameters Based on Gradient Descent Method. The method of parameter training in BP neural network generally adopts the method of gradient descent. Gradient descent method is used to find the minimum value of the function, and it is an iterative algorithm, which has been widely used in machine learning. Gradient is a vector, the direction of the derivative is a scalar, the direction of the gradient refers to the direction of the maximum direction of the derivative, and gradient modulus is equal to the maximum value of the directional derivative. Solving the gradient is to find the partial derivative for each independent variable, and then the partial derivative as the direction of the independent variable coordinates can be

$$\nabla J(\omega, a, \beta, b) = \left(\frac{\partial J(\omega, a, \beta, b)}{\partial \omega_{ij}}, \cdots, \frac{\partial J(\omega, a, \beta, b)}{\partial a_j}, \cdots, \frac{\partial J(\omega, a, \beta, b)}{\partial \beta_{jo}}, \cdots, \frac{\partial J(\omega, a, \beta, b)}{\partial b_o}\right) \tag{7}$$

Firstly, select an initial point and calculate the gradient of the point and then update the independent variable in the direction of the gradient. If the k-th iteration value of ω_{ij} is $\omega_{ij}^{(k)}$, then

$$\omega_{ij}^{(k+1)} = \omega_{ij}^{(k)} - \eta \nabla_{\omega} J\left(\omega_{ij}^{(k)}, a, \beta, b\right) \tag{8}$$

where η is called the step length or the learning rate, which indicates the size of the argument in each iteration step.

In the same way, other parameters in the network iteratively update the expression

$$a_j^{(k+1)} = a_j^{(k)} - \eta \nabla_a f\left(\omega, a_j^{(k)}, \beta, b\right) \tag{9}$$

$$\beta_{jo}^{(k+1)} = \beta_{jo}^{(k)} - \eta \nabla_{\beta} f\left(\omega, a, \beta_{jo}^{(k)}, b\right) \tag{10}$$

$$b_o^{(k+1)} = b_o^{(k)} - \eta \nabla_b f\left(\omega, a, \beta, b_o^{(k)}\right) \tag{11}$$

Note 1. The gradient descent method slows down when approaching the minimum value. There is no standardized way to set the iteration step size of the algorithm. The algorithm may fall into the local optimum and may not be able to obtain the global minimum. It is difficult to update the parameters in the neural network adaptively with the increasing of input samples.

3.3. Estimation of Evaluation Function Parameters Based on Extreme Learning Machine Method. Extreme Learning Machine (ELM) is an algorithm proposed by Professor Huang Guangbin from Nanyang Technological University to solve neural networks. The biggest characteristic of ELM compared to the traditional neural network, especially single hidden neural network, is faster than the traditional learning algorithm.

When we randomly initialize ω and a in BP neural network and set the bias vector b to zero and consider only β, the BP neural network can be regarded as an ELM [14]. The matrix representation of the ELM combined with the above equation (2) is as follows:

$$Z \cdot \beta = Y \tag{12}$$

where Z is the output of the hidden layer node, β is the output weight, and Y is the expected output.

Once the input weight ω_{ij} and the hidden layer bias a_j are given randomly in the ELM algorithm, the output matrix Z of the hidden layer is uniquely determined. A single hidden layer neural network that had been trained can be transformed into solving a linear system: $Z \cdot \beta = Y$. Then the problem of solving the hidden layer output weight vector b is transformed into the least square problem [15].

Note 2. Single hidden layer neural network in the process of iterative algorithm training all the parameters in the network should be retrained, so its training speed is slower. If using extreme learning machine to solve the network parameters, the training speed of the network will be greatly improved. The BP neural network has been reduced to ELM to reduce the computational complexity. However, ω and a are randomly initialized in the network, which leads to unstable output of the network.

Least squares method is a mathematical optimization technique that finds the best match for the data by minimizing the square of the error. The least squares method can be used to find the unknown data easily and minimize the sum of squares of the errors between the obtained data and the

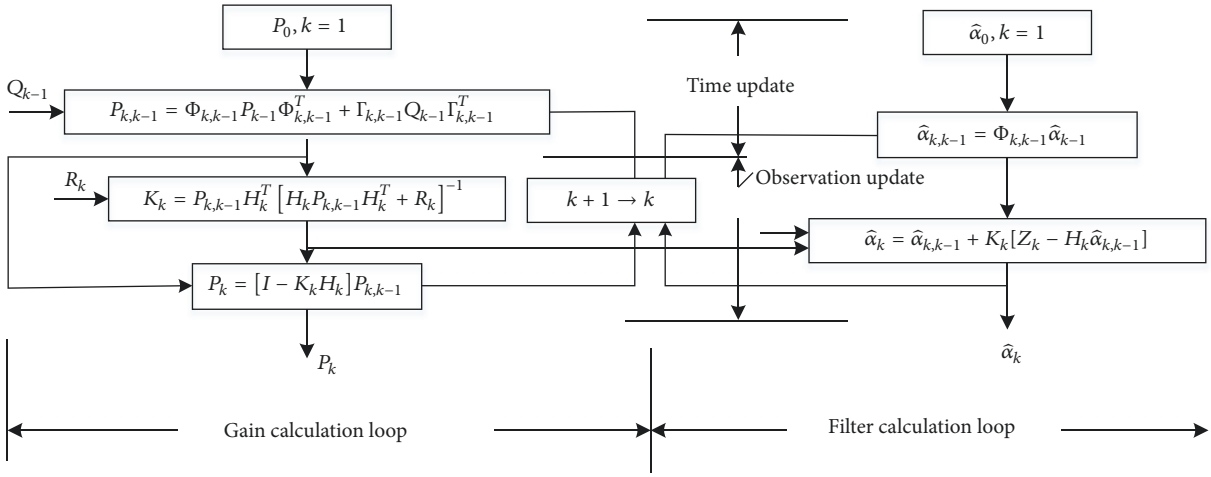


Figure 3: The process of Kalman filtering.

actual data. For the linear model established in (11), for the least squares, the final matrix representation is as follows:

$$\min \left\| Z\beta - Y \right\|_2 \tag{13}$$

The final optimal solution is

$$\beta = \left(Z^T Z \right)^{-1} Z^T Y \tag{14}$$

Note 3. The least squares method is a centralized approach that is based on the whole and cannot adjust and update the network parameters adaptively as the sample data increases.

According to the research, Kalman filtering possesses the potential to adjust parameters in the network in real time as the number of samples increases or modalities update.

3.4. Estimation of Evaluation Function Parameters Based on Kalman. If we want to use Kalman filtering to update the network parameters in real time, then we must establish the state equation and the measurement equation that fit the Kalman filter [16, 17]. Considering that the parameters β and b to be estimated are all subject to a certain degree of random disturbance, we model the state equation of Kalman filter as follows:

$$\alpha (k+1) = \alpha (k) + w (k) \tag{15}$$

In the above formula, $\alpha = [\beta, b]^T$. To simulate the interference to be estimated, we added the white noise sequence $w(k)$ to the equation.

In the above BP neural network algorithm derivation process, (3) can be seen as Kalman filter measurement equation

$$y (k) = \sum_{j=1}^{l} \beta_{jo} f \left(\sum_{i=1}^{n} \omega_{ij} x_i (k) + a_j \right) + b + v (k) \tag{16}$$

Considering that we are building a linear Kalman filter model, we randomly initialize w and e in the above formula, so the above equation is distorted to

$$y (k) = H (k) \alpha + v (k) \tag{17}$$

where $H(k) = [g(\sum_{i=1}^{n} \omega_{i1} x_i(k) + a_j), g(\sum_{i=1}^{n} \omega_{i2} x_i(k) + a_j), \cdots, g(\sum_{i=1}^{n} \omega_{ij} x_i(k) + a_j), 1]$; since the parameter α to be estimated is slowly changing, we also added the white noise sequence $w(k)$ to the model.

Therefore, the establishment of Kalman filter model is as follows:

$$\begin{aligned} \alpha (k+1) &= A (k+1, k) \alpha (k) + w (k) \\ y (k) &= H \alpha (k) + v (k) \end{aligned} \tag{18}$$

In the Kalman filter model, both the process noise $w(k)$ and the observed noise $v(k)$ are white noise sequences, which are constant values during the sampling interval. With $E\{w(k)w'(k)\} = Q$ and $E\{v(k)v'(k)\} = R$, and when $w(k)$ and $v(k)$ are independent of each other, $E\{w(k)v'(k)\} = 0$. The process of Kalman filtering is shown in Figure 3. Now Kalman optimal filtering equation is summarized as follows:

Time update is

$$\begin{aligned} \widehat{\alpha} (k+1 \mid k) &= A (k+1, k) \widehat{\alpha} (k \mid k) \\ &\quad + B (k+1, k) u (k) \end{aligned} \tag{19}$$

$$\begin{aligned} P (k+1 \mid k) &= A (k+1, k) P (k \mid k) A^T (k+1 \mid k) \\ &\quad + \Gamma (k+1, k) Q (k) \Gamma^T (k+1, k) \end{aligned} \tag{20}$$

Measurement update is

$$\begin{aligned} K (k+1) &= P (k+1 \mid k) H^T (k+1) \\ &\quad \cdot \left[H (k+1) P (k+1 \mid k) H^T (k+1) \right. \\ &\quad \left. + R (k+1) \right]^{-1} \end{aligned} \tag{21}$$

$$\begin{aligned}\widehat{\alpha}(k+1 \mid k+1) &= \widehat{\alpha}(k+1 \mid k) + K(k+1) \\ &\cdot [C(k+1)\,\widetilde{\alpha}(k+1 \mid k) + v(k+1)]\end{aligned} \tag{22}$$

$$\begin{aligned}P(k+1 \mid k+1) &= P(k+1 \mid k) - P(k+1 \mid k)\,H^T(k \\ &+ 1) \cdot \left[H(k+1)\,P(k+1 \mid k)\,H^T(k+1)\right. \\ &\left. + R(k+1)\right]^{-1} \cdot H(k+1)\,P(k+1 \mid k)\end{aligned} \tag{23}$$

In the above five formulas, (19) to (23), $\widehat{\alpha}(k+1 \mid k)$ denotes the state estimation value at time $k+1$; $P(k+1 \mid k)$ denotes the covariance matrix corresponding to $\alpha(k+1 \mid k)$; $K(k+1)$ denotes the optimal gain matrix at time $k+1$; $\widetilde{\alpha}(k+1 \mid k)$ denotes the state estimation error at time $k+1$, $\widetilde{\alpha}(k+1 \mid k) = \alpha(k+1) - \widehat{\alpha}(k+1 \mid k)$; $\widehat{\alpha}(k+1 \mid k+1)$ is the optimal state estimate at time $k+1$; and $P(k+1 \mid k+1)$ represents the covariance matrix corresponding to $\alpha(k+1 \mid k+1)$.

After the above Kalman filter model is established, the observation matrix H is solved with the parameters in the trained neural network, and then the hidden layer output weight vector and the bias vector α are updated according to the Kalman filter algorithm.

Note 4. If we consider ω and a in $f(\sum_{i=1}^{n} \omega_{ij} x_i(k) + a_j)$, the measurement equation of Kalman filter we have established becomes a nonlinear equation. The equation of state is a column vector $[\omega, a, \beta, b]^T$ consisting of ω, a, β, and b. Nonlinear filtering method for state estimation will greatly increase the complexity of the algorithm [18, 19].

4. Feature Weight Calculation and Fault Diagnosis

Mean Impact Value (MIV) is used to evaluate the importance of each independent variable for the dependent variable size [20]. MIV is an indicator used to determine the magnitude of the effect of input neurons on the output neurons, whose symbols represent the relative direction, and the absolute value represents the relative importance of the effect. In a word, the greater the feature weight calculated by the evaluation function, the more sensitive the output of the function to the change of the characteristic variable. The feature weighting used for sample attributes is mainly to amplify the changes of key feature variables, making the output more sensitive to changes. The specific calculation process is as follows:

$$X_{\pm\delta}^{(i)} = \left[x_{\pm\delta}^{(i)}(1), x_{\pm\delta}^{(i)}(2), \cdots, x_{\pm\delta}^{(i)}(L)\right] \tag{24}$$

$$x_{\pm\delta}^{(i)}(k) = \left[x_1(k), \cdots, (1 \pm \delta)\,x_i(k), \cdots, x_n(k)\right]^T \tag{25}$$

$$Z_j = f\left(\sum_{i=1}^{n} \omega_{ij} x_{\pm\delta}^{(i)}(k) + a_j\right) \tag{26}$$

$$\widehat{Y}_{i,\pm} = \sum_{k=1}^{L} Z_j \beta_{jo} + b_o \tag{27}$$

$$\begin{aligned}IV_i &= \widehat{Y}_{i,+} - \widehat{Y}_{i,-} \\ &= F(x_1(k), \cdots, (1+\delta)\,x_i(k), \cdots, x_n(k)) \\ &\quad - F(x_1(k), \cdots, (1-\delta)\,x_i(k), \cdots, x_n(k))\end{aligned} \tag{28}$$

Let $\delta \cdot x_i(k) = \Delta x_i$; then (28) becomes

$$\begin{aligned}IV_i &= \widehat{Y}_{i,+} - \widehat{Y}_{i,-} \\ &= F(x_1(k), \cdots, x_i(k) + \delta \cdot x_i(k), \cdots, x_n(k)) \\ &\quad - F(x_1(k), \cdots, x_i(k) - \delta \cdot x_i(k), \cdots, x_n(k)) \\ &= F(x_1(k), \cdots, x_i(k) + \Delta x_i, \cdots, x_n(k)) \\ &\quad - F(x_1(k), \cdots, x_i(k) - \Delta x_i, \cdots, x_n(k))\end{aligned} \tag{29}$$

In summary, take $0.1 \le \delta \le 0.3$, $i = 1, 2, \cdots, n$. After the network training is terminated, the training samples $X_{+\delta}^{(i)}$ and $X_{-\delta}^{(i)}$ are simulated as the simulation samples using the built-up network to simulate, respectively. Getting two simulation results $\widehat{Y}_{i,+}$ and $\widehat{Y}_{i,-}$, IV_i are the average influence of the i-th variable in the sample data. Similarly, the average effect of the other variables in the sample data can be obtained.

$$IV = \left[IV_1, IV_2, \cdots, IV_n\right]^T \tag{30}$$

The parameters in IV are the value of influence for feature variables in sample data to the output. Finally, the IV value of m group is simulated, and the average value of the corresponding mean IV is to be taken as MIV.

$$MIV = \frac{1}{m}\sum_{i=1}^{m} IVi \tag{31}$$

The magnitude of the absolute value of MIV is the relative importance of the influence of the respective variables on the network output, thus achieving the weighting of the sample data characteristics, and then the PCA algorithm is used for fault diagnosis.

5. Simulation of the Algorithm

5.1. Simulation Data. The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables, temperature (T), ambient pressure (AP), relative humidity (RH), and exhaust vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. A Combined Cycle Power Plant (CCPP) is composed of gas turbines (GT), steam turbines (ST), and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the vacuum is collected from and has effect on the steam turbine, the other three of the ambient variables affect the GT performance [21].

In this experiment, 300 sets of data T were used in the simulation test. In the evaluation function establishment

Table 1: Neural network fitting results before and after the parameter update.

Original Data T	The error rate algorithm fitting (%)			
	0.99T	1.00T	1.02T	1.05T
BPNN	1.33	0.88	2.16	4.86
Kalman-BPNN	0.88	0.89	0.93	1.01

Table 2: The characteristic weights of the sample data.

Original Data T	Weight							
	Original weight				Updated weight			
0.99T	-10.76	-0.59	0.10	-3.21	-7.46	-1.98	0.97	-1.06
1.00T	-10.76	-0.59	0.10	-3.21	-7.54	-2.00	0.98	-1.07
1.02T	-10.76	-0.59	0.10	-3.21	-7.69	-2.03	1.00	-1.09
1.05T	-10.76	-0.59	0.10	-3.21	-7.91	-2.09	1.03	-1.12

and fault diagnosis process, 200 sets were selected as the training data and 100 sets as the test data. To simulate the variation of the system modal, we add an interference $r = 2 + 5 * randn(k, 1)$ to the third feature input of the original dataset, where k is the number of samples. We multiply the original sample data output T by 0.99, 1.02, and 1.05 to simulate changes in the system modality and then test the fitting output of the BP neural network before and after the parameter update. The error rate is used to characterize the fitting accuracy of the model.

Error rate is

$$s\_e = \frac{\sqrt{\sum_{i=1}^{L} (y_i - \hat{y}_i)^2 / n}}{\overline{y}} \times 100\% \tag{32}$$

5.2. Discussion. For the data before and after the system modal change, we use the original trained neural network and the neural networks of hidden layer output weights were updated by Kalman filter algorithm to fit the data, and then we calculated the error rate to characterize the fitting accuracy, respectively. Combined with Figure 4 and Table 1, we can clearly see that the algorithm proposed in this paper real-time update of the model parameters can be very good to ensure the accuracy of the model.

After the traditional neural network parameter training is completed, if the modality of the original system changes, we often retrain the entire network. However, we all know that neural network parameter training process is time-consuming. For those systems whose modality is constantly changing, it is obviously unreasonable to adopt a method of retraining the entire network parameters. The Kalman filtering-based real-time update algorithm for the hidden layer output weights of the BP neural network solves the above problems. On the basis of not changing the other training parameters of the original network, the Kalman filtering model is established, and then the hidden layer output parameters of the BP neural network are updated according to the new data. It can be seen from Table 1 and Figure 4 in this paper that the greater the change in the modality of the original system, the greater the network fitting error. Among the several modal changes we simulated, the fitting errors were 1.33%, 0.88%, 2.16%, and 4.86%, respectively. It can be seen that the change of the original system modality and the original network can no longer trace the output well. However, in our simulation of several modal changes, using the proposed BP neural network local parameter update algorithm to update the hidden layer output weights in the network, the fitting error is 0.88%, 0.89%, 0.93%, and 1.01%, respectively. It can be seen that using the proposed BP neural network local parameter real-time updating algorithm greatly ensures the accuracy of the model, which is the premise to ensure the validity of the evaluation function.

Table 2 is the vector of weights obtained before and after updating the network parameters. We use the weight vectors obtained before and after updating the network parameters to weight the original sample data and then using the PCA fault diagnosis algorithm to do the fault diagnosis. Finally, the fault diagnosis results of the two kinds of weighted data are compared with the results of failure diagnosis of the sample data without weighting. As shown in Table 3 and Figure 5, the three are significantly different in the SPE statistics. After the modal changes, the weighted weights using the original weights are lower than the unweighted diagnostic accuracy. The weighted weights using the updated weights are significantly higher than the unweighted diagnostic accuracy, with the maximum increase of 18%.

It can be seen from the above simulation data and simulation plot that using Kalman filter algorithm to update the hidden layer output weights of neural networks in real time is feasible. After the observation data is updated, the Kalman filter improves the state quantity with new observation data to get the state quantity at this moment, and the real-time performance of the algorithm is greatly improved. When the system modal changes, the output of the model can be well fitted to the true value without the need of retraining all the parameters of the neural network. At the same time, an accurate evaluation function to obtain the characteristic weight effectively enhances the diagnostic performance of the fault diagnosis algorithm. However, since the method eventually simplifies the model to a linear model and updates some parameters in the network, it is worthwhile to study

Table 3: Accuracy of fault diagnosis for different weights.

Original Data T	Accuracy of fault diagnosis (%)					
	Unweighted		Original weight		Updated weight	
	T^2_{99}	SPE	T^2_{99}	SPE	T^2_{99}	SPE
0.99T	98	50	4	42	98	66
1.00T	100	78	100	100	100	84
1.02T	98	50	4	42	98	68
1.05T	98	50	4	42	98	68

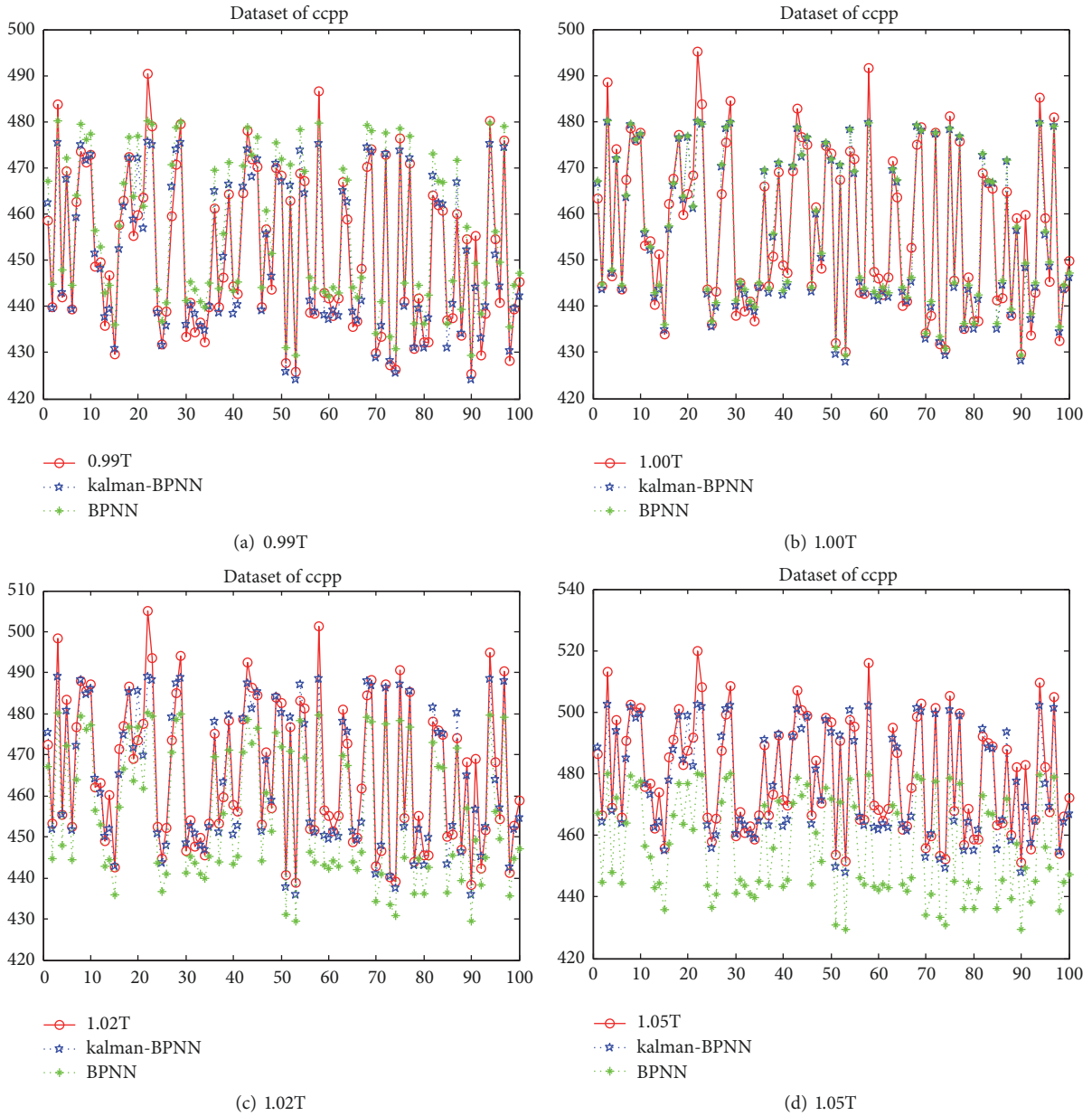
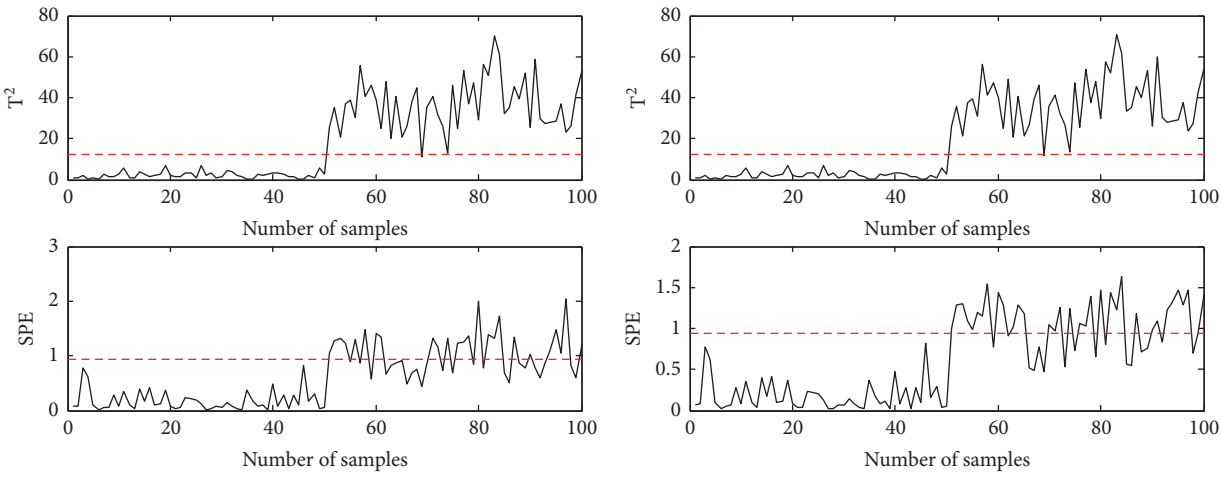
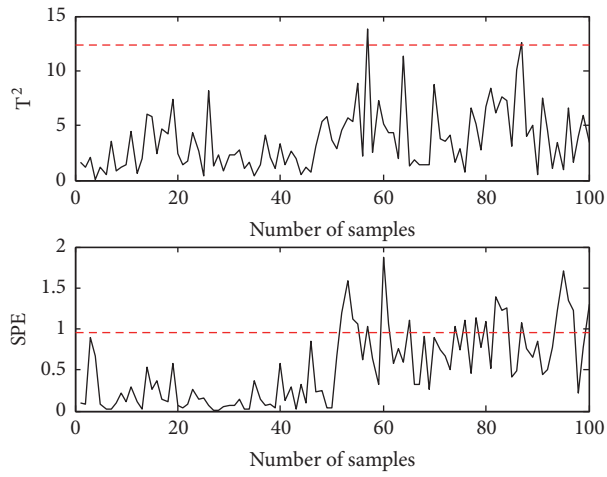


Figure 4: Neural network fitting output before and after the parameters of neural network were updated by Kalman filter after system modal change.



(a) Unweighted T ^ 2 and SPE plotted

(b) The updated weight weighted T ^ 2 and SPE plotted



(c) Original weight weighted T ^ 2 and SPE plotted

Figure 5: Fault detection diagram for 1.05T.

all parameters of the network using some nonlinear filtering algorithms [22].

6. Conclusions

This paper applies the Kalman filter to the real-time update of the hidden layer output weights of BP neural network, which greatly improves the applicability and real-time performance of the BP neural network and ensures the accuracy of the model parameters, so that the obtained evaluation function is accurate. This also makes it possible to characterize the impact of each feature variable on the output based on the weights of the data features obtained from the evaluation function. When there is not big change in the system state, the output of BP neural network hidden layer is adjusted by Kalman filter to compensate the output of the system, in order to avoid retraining the network once the system modality changes. Retraining will result in a waste of resources and may not be tolerated even with frequent retraining of some system models. The simulation results on a standard UCI-ccpp dataset validate the effectiveness of the proposed algorithm.

Once the training of neural network is completed, it is always a difficult problem to retrain global parameters and update local parameters after it is applied to practical industrial systems. The proposed algorithm only completes the real-time update of the hidden layer output weights of the BP neural network without changing other parameters in the network. On this basis, in the next step we are going to combine some nonlinear filtering algorithms and contents of incremental learning to seek a breakthrough in the regulation and updating of global network parameters.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation (NNSF) of China under Grants nos. U1509203, 61490701, 61573137, 61673160, and 61333005.

References

[1] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 16, no. 1, pp. 24–38, 2005.

[2] T.-Y. Kwok and D.-Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 8, no. 5, pp. 1131–1148, 1997.

[3] S. Ganjefar and M. Tofighi, "Optimization of quantum-inspired neural network using memetic algorithm for function approximation and chaotic time series prediction," *Neurocomputing*, vol. 291, pp. 175–186, 2018.

[4] Y. W. Wong, K. P. Seng, and L.-M. Ang, "Radial basis function neural network with incremental learning for face recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 4, pp. 940–949, 2011.

[5] L. Fu, "Incremental knowledge acquisition in supervised learning networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 26, no. 6, pp. 801–809, 1996.

[6] K. W. Kow, Y. W. Wong, R. K. Rajkumar, R. K. Rajkumar, and D. Isa, "Incremental unsupervised learning algorithm for power fluctuation event detection in PV grid-tied systems," *Lecture Notes in Electrical Engineering*, vol. 398, pp. 673–679, 2017.

[7] H. Wen, W. Xie, J. Pei, and L. Guan, "An incremental learning algorithm for the hybrid RBF-BP network classifier," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, 2016.

[8] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 2, no. 3, pp. 302–309, 1991.

[9] S. Wu and M. J. Er, "Dynamic fuzzy neural networks—a novel approach to function approximation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 2, pp. 358–364, 2000.

[10] S. Q. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 4, pp. 578–594, 2001.

[11] M. Zhang, X. Ma, B. Qin et al., "Information fusion control with time delay for smooth pursuit eye movement," *Physiological Reports*, vol. 4, no. 10, Article ID e12775, 2016.

[12] L.-Z. Xu, X.-L. Feng, and C.-L. Wen, "Sequential fusion filtering for networked multi-sensor systems based on noise estimation," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 42, no. 1, pp. 160–168, 2014.

[13] S. Ding and Q.-H. Wu, "A MATLAB-based study on approximation performances of improved algorithms of typical BP neural networks," *Applied Mechanics and Materials*, vol. 313-314, pp. 1353–1356, 2013.

[14] G. Huang, L. Chen, and C. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 17, no. 4, pp. 879–892, 2006.

[15] S. Rhode, K. Usevich, I. Markovsky, and F. Gauterin, "A recursive restricted total least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 21, pp. 5652–5662, 2014.

[16] W. Chenglin, *Multi-Scale Dynamic Modeling Theory and Its Applications*, Science Press, 2008.

[17] Q. Liu, Z. Wang, X. He, and D. Zhou, "On Kalman-consensus filtering with random link failures over sensor networks," *IEEE Transactions on Automatic Control*, vol. 99, pp. 1-1, 2017.

[18] Q. Liu, Z. Wang, X. He, G. Ghinea, and F. E. Alsaadi, "A resilient approach to distributed filter design for time-varying systems under stochastic nonlinearities and sensor degradation," *IEEE Transactions on Signal Processing*, vol. 65, no. 5, pp. 1300–1309, 2017.

[19] C. Wen, Z. Wang, J. Hu, Q. Liu, and F. E. Alsaadi, "Recursive filtering for state-saturated systems with randomly occurring nonlinearities and missing measurements," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 5, pp. 1715–1727, 2018.

[20] S. Ji, X. Xu, and C. Wen, "A kind of K—Nearest neighbor fault diagnosis method based on MIV data transformation," in *Proceedings of the 2017 Chinese Automation Congress (CAC)*, pp. 6306–6310, Jinan, October 2017.

[21] M. Lichman, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2013, http://archive.ics.uci.edu/ml.

[22] C. Wen, Z. Wang, T. Geng, and F. E. Alsaadi, "Event-based distributed recursive filtering for state-saturated systems with redundant channels," *Information Fusion*, vol. 39, pp. 96–107, 2018.