1. Detailed Methodology of PCA

Assuming that the number of samples contained in the experimental data is m, and the number of indicators of each sample is n, then forming an $m \times n$ dimensional matrix, and then calculate the covariance matrix R according to the standardized data:

$$R = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{n1} & \cdots & r_{nn} \end{bmatrix} \tag{S1}$$

Calculating the eigenvalues of the covariance matrix R, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, corresponding eigenvectors are $v_1, v_2, \cdots, v_n$, and $v_j = (v_1, v_2, \cdots, v_n)^{\mathrm{T}}$, v_j represents the n_{th} component of the j_{th} eigenvector. Then the new index variables are reconstructed by the eigenvectors:

$$\begin{cases} f_1 = v_{11}X_1{}^* + v_{21}X_2{}^* + \cdots + v_{n1}X_n{}^* \\ f_2 = v_{12}X_1{}^* + v_{22}X_2{}^* + \cdots + v_{n2}X_n{}^* \\ \vdots \\ f_n = v_{1n}X_1{}^* + v_{2n}X_2{}^* + \cdots + v_{nn}X_n{}^* \end{cases} \tag{S2}$$

In Equation (S2), f_1 is the first principal component, f_2 is the secondary principal component, $\cdots$, f_n is the n principal component.

Calculating the contribution rate of every principal component p_j (j = 1,2, $\cdots$, n), and the cumulative contribution rate is α_l ($l \leq n$):

$$p_j = \lambda_j / \sum_{k=1}^{n} \lambda_k \tag{S3}$$

$$\alpha_l = \sum_{k=1}^{l} \lambda_k \,/ \sum_{k=1}^{n} \lambda_k \tag{S4}$$

In this research, the cumulative contribution rate is about 90.0%, which can not only better retain the original data information, but also reduce the computational complexity.

2. Detailed Methodology of GA

The set of multiple given initial weight matrices as a population, and each initial weight matrix in the set is an individual. Firstly, the population needs to be initialized, and the binary form coding method is used to encode these individuals which only represents the weight and threshold of the BP neural network. The encoding length of the individual can be calculated as Equation (S5):

$$N_w + N_c = \sum_{i=1}^{N_h} L_i \cdot L_{i+1} + \sum_{j=1}^{N_h+1} L_j \tag{S5}$$

where N_w and N_c is the total number of link-weight and threshold, respectively.

The fitness function is used to evaluate the fitness of an individual. The quality of the fitness function directly affects the performance of GA. Its setting should be designed according to the actual problem requirements. Hence, the appropriate equation is chosen to solve the error in this article. the fitness function is defined as the sum of the absolute errors between the expected output and the predicted output of the BP neural network. The smaller the fitness function value, the more accurate the training and the better the prediction accuracy. The equation is shown as follows:

$$F = \sum_{j=1}^{N} \sum_{i=1}^{m} \left| y_{ij} - y_{ij}^* \right| \tag{S6}$$

This operation can be used to select the individual with high fitness in the population through the roulette method in this work. The ideal individual can be selected by a certain probability in Equation (S7):

$$S_i = f_i / \sum_{j=1}^{C} f_j \tag{S7}$$

where S_i is the selection probability of the individual i. f_i equals to $1/F_i$, and F_i is the value of fitness. C is the population size. The larger f_i is a preferred individual to be selected.

The crossover operator is the core operator of GA, which determines the global searching ability. The mutation operator is the auxiliary operator of GA, which determines the local ability of searching.

Through the repetition of calculation of fitness function value, the operator selection, the crossover operator selection and the mutation operator selection, until the optimal population is generated.

3. Detailed Methodology of BP

$$b_j = \sigma(\sum_{i=1}^{n} \omega_{ij} X_i \theta_i) \tag{S8}$$

where $\sigma(x)$ is a hierarchical transfer function between neurons, ω_{ij} is the weight between the input layer and the hidden layer, X_i is neuron node between input layer, b_j is the value of hidden layer neuron, θ_i is the offset vector between the input layer and the hidden layer.

$$y_k = \sigma(\sum_{i=1}^{n} \varphi_{kj} b_j \mu_k) \tag{S9}$$

where y_k is the actual value of output layer, φ_{kj} is the weight between the output layer and the hidden layer, μ_k is the offset vector between the hidden layer and output layer.

Calculating the error of the output value of the BP neural network, which determines whether forward propagation is required. In the signal forward propagation step, the sum of training error for samples can be shown as Equation (S10):

$$E = \sum_{j=1}^{N} \sum_{i=1}^{m} (y_{ij} - y_{ij}^{*})^2 \tag{S10}$$

where y_{ij}^{*} is the actual value of output layer, y_{ij} is the expected value.

If the sum of training error for samples does not meet the preset threshold, forward propagation is required. Forward propagation requires adjusting the data from the output layer to the input layer, using the error function to solve the weights φ_{kj} of output layer.

$$\Delta\varphi_{kj} = \frac{\partial E}{\partial \varphi_{kj}} \tag{S11}$$

$$\Delta\mu_k = \frac{\partial E}{\partial \mu_k} \tag{S12}$$

$$\Delta\omega_{ij} = \frac{\partial E}{\partial \omega_{ij}} \tag{S13}$$

$$\Delta\theta_i = \frac{\partial E}{\partial \theta_i} \tag{S14}$$

where $\Delta\varphi_{kj}$ represents output layer weight change value, $\Delta\mu_k$ represents the offset vector change value between the hidden layer and output layer, $\Delta\omega_{ij}$ represents input layer weight change value, and $\Delta\theta_i$ represents the offset vector change value between the input layer and the hidden layer.

Repeating the above calculation of change value until the sum of error reaches the threshold or the number of iterations reaches the maximum value, and the BP neural network algorithm will output the result. With the grow of the number of layers, the complexity of the calculation will increase and the error will decrease, but when the local search algorithm or gradient descent algorithm is used internally, it is easy to cause the algorithm to fall into the local minimum value, thereby reducing the convergence speed of the algorithm. Hence the number of hidden layer nodes k is shown as:

$$k = \sqrt{m+n} + b \tag{S15}$$

where m is the number of nodes in the input layer, n is the number of nodes in the output layer, and $b \in [0,10]$.

The training process was shown as Figure S1.

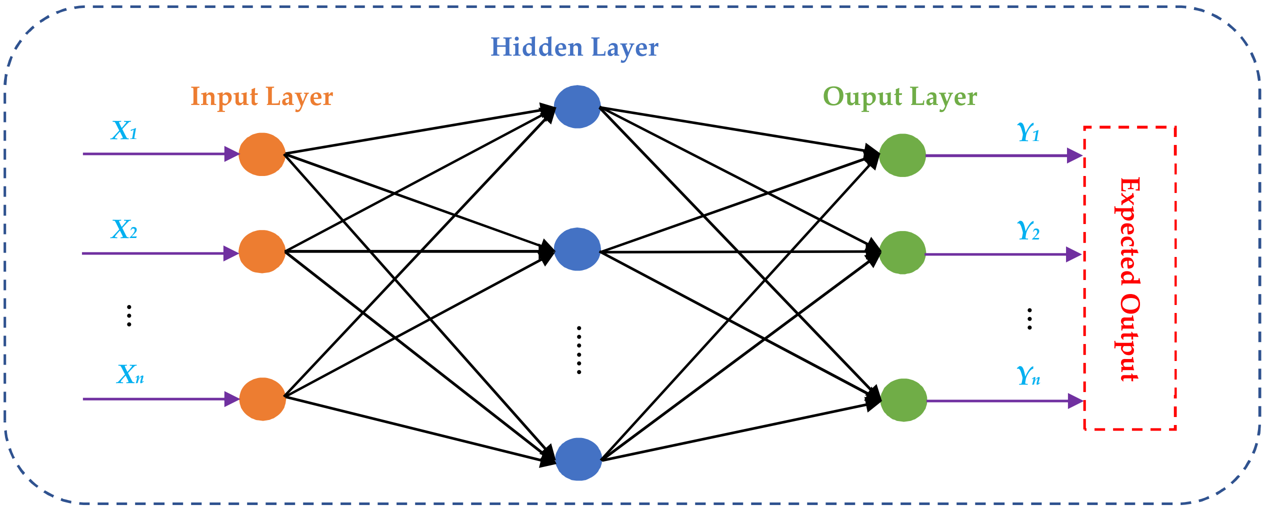

Figure S1. Structure of BP neural network.

The flow chart of the PCA-GA-BP neural network algorithm was shown in Figure S2.

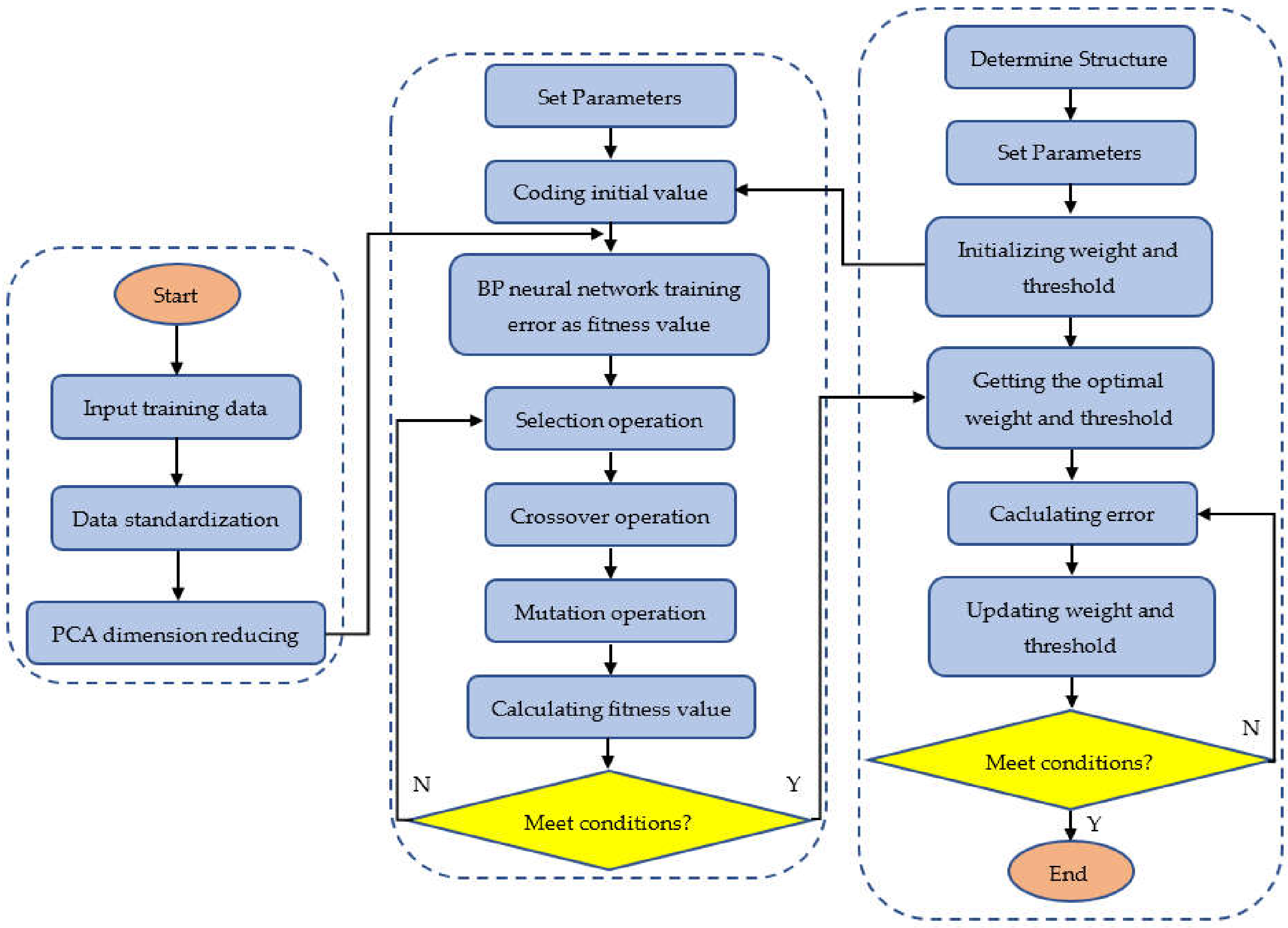

Figure S2. The flow chart of the PCA-GA-BP neural network algorithm.