Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: 0005-1144 (Print) 1848-3380 (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

Hybrid intelligent deep kernel incremental extreme learning machine based on differential evolution and multiple population grey wolf optimization methods

Di Wu, Zong Shun Qu, Feng Jiao Guo, Xiao Lin Zhu & Qin Wan

To cite this article: Di Wu, Zong Shun Qu, Feng Jiao Guo, Xiao Lin Zhu & Qin Wan (2019) Hybrid intelligent deep kernel incremental extreme learning machine based on differential evolution and multiple population grey wolf optimization methods, Automatika, 60:1, 48-57, DOI: 10.1080/00051144.2019.1570642

To link to this article: https://doi.org/10.1080/00051144.2019.1570642

Published online: 10 Feb 2019.

AUTOMATIKA
2019, VOL. 60, NO. 1, 48–57
https://doi.org/10.1080/00051144.2019.1570642

REGULAR PAPER

OPEN ACCESS Check for updates

Hybrid intelligent deep kernel incremental extreme learning machine based on differential evolution and multiple population grey wolf optimization methods

Di Wu[a,b], Zong Shun Qu[a], Feng Jiao Guo[a], Xiao Lin Zhu[a] and Qin Wan[a]

[a]College of Electrical and Information Engineering, Hunan Institute of Engineering, Xiang Tan, People's Republic of China; [b]Hunan Province Cooperative Innovation Center for Wind Power Equipment and Energy Conversion, Hunan Institute of Engineering, Xiang Tan, People's Republic of China

ABSTRACT

Focussing on the problem that redundant nodes in the kernel incremental extreme learning machine (KI-ELM) which leads to ineffective iteration increase and reduce the learning efficiency, a novel improved hybrid intelligent deep kernel incremental extreme learning machine (HI-DKIELM) based on a hybrid intelligent algorithms and kernel incremental extreme learning machine is proposed. At first, hybrid intelligent algorithms are proposed based on differential evolution (DE) and multiple population grey wolf optimization (MPGWO) methods which used to optimize the hidden layer neuron parameters and then to determine the effective hidden layer neurons number. The learning efficiency of the algorithm is improved by reducing the network complexity. Then, we bring in the deep network structure to the kernel incremental extreme learning machine to extract the original input data layer by layer gradually. The experiment results show that the HI-DKIELM methods proposed in this paper with more compact network structure have higher prediction accuracy and better ability of generation compared with other ELM methods.

ARTICLE HISTORY
Received 24 December 2017
Accepted 14 January 2019

KEYWORDS
Extreme learning machine (ELM); kernel incremental extreme learning machine (KIELM); differential evolution (DE); multiple population grey wolf optimization methods (MPGWO); hybrid intelligence (HI)

1. Introduction

The artificial neural network analyses the data through the abstract simulation process to the biological neuron network, thereby realizing some functions such as data classification, system identification, function approximation and numerical estimation. However, the training efficiency and learning ability of the traditional Single Hidden Layer Feed Forward Neural Networks (SLFNS) is still too low. It is need to update all parameters of the network in the learning process. Recently, Huang et al. [1] proposed an extreme learning machine (ELM) algorithm for training single hidden layer feed forward neural networks. Compared to the traditional neural network, the parameters of the hidden layer nodes in ELM are randomly initiated and then fixed without iteratively tuning and tedious iterative process. The only free parameters need to be learned are the connections or weights between the hidden layer and output layer, and its output weight is obtained by the generalized inverse solution of the matrix using regularized least squares methods. In this manner, ELM can achieve good universal approximation capability as well as high running efficiency based on excellent network learning performance and network structure, thereby, avoiding the local minimum and slow convergence problems.

In practice, because of the complexity of various problems, several methods for optimizing the ELM hidden nodes have been proposed to obtain a suitable network structure and size. Huang et al. [2] proposed a standard optimization method for classification problems, then Huang [3] also proved the possibility of using ELM for arbitrary multi-classification problems and obtained good experiment performance. In [4], the class weight has been introduced to solve the complex unbalanced learning problems to further improve the performance. At present, ELM has been widely used in face recognition, speech recognition, licence plate recognition, power system [5–7] and so on. For the reason of more classification labels, lack of training samples and insufficient feature descriptions, the recognition accuracy of ELM in traditional classification problems is undesirable. Therefore, under the precise of ensure the superiority of fast training speed and good generalization performance, which further improve the overall classification performance and recognition accuracy of ELM, is becoming the present research focus.

In traditional ELM, the higher dimensional network structures have always been used for the purpose of obtaining stronger learning ability, but the number

CONTACT Di Wu wudi6152007@163.com College of Electrical and Information Engineering, Hunan Institute of Engineering, Xiang Tan 411104, People's Republic of China; Hunan Province Cooperative Innovation Center for Wind Power Equipment and Energy Conversion, Hunan Institute of Engineering, Xiang Tan 411104, People's Republic of China

of optimal hidden layer nodes and the scale of control model are difficult to determine. For this reason, Huang et al. [8] proposed the Incremental Extreme Learning Machine (I-ELM), where the hidden nodes are added incrementally and the output weights are determined analytically. In [9], a state-of-art learning algorithm known as Enhanced Incremental Extreme Learning Machine (EI-ELM) is presented to elect the effective hidden layer node to construct the network structure using some novel optimization algorithms while reducing the complexity of the network to some extent. However, when the size of the network is too large, the iteration of the EI-ELM is greatly increased, which affects the generalization ability. Huang et al. [10] proposed the Barron-optimized Convex Incremental Extreme Learning Machine (CI-ELM) to calculate the output weights of existing nodes again after the increase of hidden layer nodes to obtain a higher convergence rate. In [11], a hybrid incremental Extreme Learning Machine (HI-ELM) is proposed using chaos optimization algorithm to optimize the parameters of hidden layer nodes. However, the current I-ELM still has some problems need to be solved urgently. The complexity of the network structure will be increasing due to the reason of redundant nodes which reduces the learning efficiency. The convergence rate is low; the number of hidden layer nodes exceeds the number of learning samples. More sensitive to the new data, the online prediction ability is not strong.

The combination of the parameters is crucial for the ELM because it is affecting the training speed and learning accuracy of the ELM to some extent. Therefore, the intelligent optimization algorithm to optimize the ELM parameters based on bionics methods for the purpose of improving the learning speed and accuracy is becoming the research focus. In [12], a differential evolution (DE) algorithm utilized to adjust the ELM input parameters is proposed. In [13], an adaptive DE algorithm to optimize the parameters of the hidden layer nodes is given, and then the MP generalized inverse method is utilized to solve the output weights. In [14], an improved particle swarm optimization algorithm is utilized to optimize hidden layer node parameters. In [15], a hybrid intelligent ELM is proposed using the DE algorithm and the particle swarm optimization method to optimize the hidden layer nodes. However, the aforementioned hybrid intelligent optimization algorithm still faces two problems: although the DE algorithm has strong global optimization ability but will appear premature convergence problems, while the particle swarm optimization algorithm can perform local optimization but is searching speed are too slow.

Meanwhile, another leading trends for hierarchical learning are called deep learning (DL), similarly, the deep architecture extracts feature by a multilayer feature representation framework, and the higher layers represent more abstract information than those from the lower ones in order to improve the ELM performance. In [16], a multilayer ELM is given which combine the excellent feature extraction capabilities of deep learning and the fast training ability of ELM. In [17], the kernel function is being introduced and a novel deep kernel ELM is proposed, and used for aero-engine component fault diagnosis to improve diagnostic accuracy.

It is noteworthy that for ELM and its variants, all of the improving algorithms are composed of two stages: at first, random feature mapping and then output weight optimization. However, for complex classification problems, the effect of using random feature mapping to boost the separability of the original sample space is often limited, which increase the dependence on subsequent output weight optimization process. Moreover, most of the current variants of ELM are based on the existing ELM framework and there are few variations for combining the ELM network structure with another network structure adjustment, except for the deep learning network.

In this paper, to ensure the superiority of the proposed network structure, a hybrid intelligent deep kernel incremental extreme learning machine is proposed in order to improve the ELM network performance. First, the deep kernel incremental ELM (DKI-ELM) is proposed based on incremental kernel ELM and the deep leaning network. And the deep network structure is used to extract the data in multiple layers to obtain effective features and improve the classification accuracy. Second, a hybrid intelligent differential evolution multiple grey wolf optimization algorithm is proposed using the global search ability of the DE algorithm and the local search capability of MPGWO algorithm in order to obtain the optimal output weights for the purpose of improving the training speed and classification accuracy of the ELM.

In this paper, our major contribution is summarized as follows:

1) An HI-DKIELM (hybrid intelligent deep kernel incremental extreme learning machine) network classifier is designed. HI-DKIELM consists of a deep learning network and kernel incremental extreme learning machine of cascade, where the input data through the deep leaning network to extract more information and boost the separability can achieve higher dimensional spatial mapping, then the ELM network can be utilized to provide a superior classification surface. In this way, the HI-DKIELM proposed in this paper combines the advantages of the deep learning network and the KIELM network, and can improve the performance effectively.
2) In order to explore an optimal parameter belonging to the Extreme Learning Machine, an appropriate hybrid intelligent optimize algorithm for HI-DKIELM is presented. The proposed hybrid

differential evolution multiple grey wolf optimization algorithm (DE-MPGWO) optimizes the method using the global search ability of the DE algorithm and the local search capability of multi-group grey wolf algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the implementation of ELM and KIELM. Section 3 presents the Hybrid Intelligent differential evolution multiple grey wolf optimization algorithm. In Section 4, the detail of the HI-DKIELM is briefly discussed. Hence, the experimental result is presented in Section 5. Section 6 concludes our work and outlines our future work to generalize the method to multibiometric recognition system.

2. Preliminary

In this part, we will give the notation of Extreme Learning Machine (ELM) and kernel incremental extreme learning machine (KI-ELM).

2.1. Extreme Learning Machine Theory

Extreme Learning Machine is a high efficient learning algorithm proposed on the singer-hidden layer neural network. Unlike other different traditional neural network, all the parameters in the Extreme Learning Machine are generated randomly and the complicated iteration process is avoided. Suppose the training set $\{x_i, t_i\}_{i=1}^{N}$ is composed of N training samples, the input is x_i which the dimension is d, t_i is the label of the output, then the output of the ELM is

$$\sum_{j=1}^{L} \beta_j g(x_i) = \sum_{j=1}^{L} \beta_j g(w_j \cdot x_i + b_j) = t_i. \quad (1)$$

In Equation (1), the parameter $w_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^{\mathrm{T}}$ is the input weight of the jth hidden node, b_j is the deviation of the jth hidden node and β_j is the weight of the jth hidden node to the output node of the ELM. $G(a_j, w_j, x_i)$ is the output function of the jth hidden node. From Equation (1), we will obtain that $h(x_i) = [G(a_1, w_1, x_i), \ldots, G(a_1, w_1, x_i)]$ is the output of hidden layer in regard to training sample x_i. And Equation (1) can be simplified as

$$H\beta = T, \quad (2)$$

where H is the hidden layer output matrix o and $h(x_i)$ is the ith row hidden layer output vector relative to the input $X_{i:}$

$$H = \begin{bmatrix} h(x_1) \\ \cdots \\ h(x_N) \end{bmatrix}_{N \times L} = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_1) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_L \cdot x_N + b_1) \end{bmatrix}. \quad (3)$$

$\beta = [\beta_1, \beta_2, \ldots, \beta_L]_{L \times m}^{T}$ is output weight matrix and $T = [t_1, t_2, \ldots, t_n]_{N \times m}^{T}$ is the expected output. In order to improve the generalization ability of ELM, a penalty factor C is introduced in Equation (3), and the output weight matrix β is

$$\beta = H^{\dagger} T = H^{\mathrm{T}} \left(\frac{1}{C} + HH^{\mathrm{T}} \right)^{-1} T. \quad (4)$$

Then the output of the extreme learning machine can be expressed as

$$f(x) = h(x)\beta = h(x) H^{\mathrm{T}} \left(\frac{1}{C} + HH^{\mathrm{T}} \right)^{-1} T. \quad (5)$$

2.2. Kernel Incremental Extreme Learning Machine (KI-ELM)

Incremental Extreme Learning Machine (I-ELM) is different from the original incremental neural network which is only a specific kind of active function can be used. I-ELM can use any continuous or piecewise continuous function as the active function. Under the equal premise learning accuracy, the training speed of the I-ELM is 1000 times faster than SVM and BP neural network. In the past 5 years, some variants of the I-ELM such as EI-ELM, PC-ELM, KI-ELM and OP-ELM have been proposed respectively. These improved Incremental Extreme Learning Machines are mainly aimed at improving the hidden layer node parameters in I-ELM. The kernel matrix of the KI-ELM can be expressed as

$$K_{\mathrm{ELM}} = HH^{\mathrm{T}} = h(x_i) \cdot h(x_j) = K(x_i, x_j), \quad (6)$$

then the output function of KI-ELM can be converted from Equation (5) to

$$f(x) = h(x)\beta = h(x) H^{\mathrm{T}} (\tfrac{1}{C} + HH^{\mathrm{T}})^{-1} T = \begin{bmatrix} K(x, x_1) \\ K(x, x_2) \\ \vdots \\ K(x, x_N) \end{bmatrix} (\tfrac{1}{C} + K_{\mathrm{ELM}})^{-1} T \quad (7)$$

In Equation (7), assuming $A = [\frac{1}{C} + K_{ELM}]$, at t moment, there are

$$A_t = \begin{bmatrix} \frac{1}{C} + K(x_1, x_1) & \cdots & K(x_1, x_N) \\ \vdots & \cdots & \vdots \\ K(x_N, x_1) & \cdots & \frac{1}{C} + K(x_N, x_N) \end{bmatrix}. \quad (8)$$

So at the $t+1$ moment, there are

$$A_{t+1} = \begin{bmatrix} \frac{1}{C} + K(x_1, x_1) & \cdots & K(x_1, x_{N+k}) \\ \vdots & \cdots & \vdots \\ K(x_{N+k}, x_1) & \cdots & \frac{1}{C} + K(x_{N+k}, x_{N+k}) \end{bmatrix}. \quad (9)$$

To simplify Equation (9), suppose

$$U_t = \begin{bmatrix} K(x_1, x_{N+1}) & \cdots & K(x_1, x_{N+k}) \\ \vdots & \cdots & \vdots \\ K(x_N, x_{N+1}) & \cdots & K(x_N, x_{N+k}) \end{bmatrix}, \quad (10)$$

$$D_t = \begin{bmatrix} \frac{1}{C} + K(x_{N+1}, x_{N+1}) & \cdots & K(x_{N+1}, x_{N+k}) \\ \vdots & \cdots & \vdots \\ K(x_{N+k}, x_{N+1}) & \cdots & \frac{1}{C} + K(x_{N+k}, x_{N+k}) \end{bmatrix}, \quad (11)$$

then Equation (9) can be simplified as

$$A_{t+1} = \begin{bmatrix} A_t & U_t \\ U_t^{\mathrm{T}} & D_t \end{bmatrix}. \quad (12)$$

Using new data, we can obtain

$$A_{t+1}^{-1} = \begin{bmatrix} A_t^{-1} + A_t^{-1} U_t C_t^{-1} U - t^{\mathrm{T}} A_t^{-1} & -C_t^{-1} U_t C_t^{-1} \\ -C_t^{-1} U_t^{\mathrm{T}} A_t^{-1} & C_t^{-1} \end{bmatrix}. \quad (13)$$

In Equation (13), $C_t = D_t - U_t^T A_t^{-1} U_t$, for testing data $X_{test} = \left[X_{test1},\ X_{test2},\ \cdots\ X_{testM}\right]$, the output value $\hat{Y}_{test}$ can be estimated online, that

$$\hat{Y}_{test} = \begin{bmatrix} K(x_{test1}, x_1) & \cdots & K(x_{test1}, x_M) \\ \vdots & \cdots & \vdots \\ K(x_{testN}, x_1) & \cdots & K(x_N, x_M) \end{bmatrix} \begin{bmatrix} A_M^{-1} y_1 \\ \vdots \\ A_M^{-1} y_M \end{bmatrix}. \quad (14)$$

3. The proposed DE-MPGWO algorithm

In this part, an improved hybrid intelligent optimized strategy called Differential Evolution Multiple Grey Wolf Optimization algorithm (DE-MPGWO) inspired from the idea of Frog Leaping algorithm (FLA) is proposed based on DE and MPGWO. In order to facilitate the proposed optimization algorithm, in Section 3.1 and Section 3.2, we briefly review the concept of DE algorithm and MPGWO algorithm, the hybrid intelligent optimization proposed in this paper is discussed in Section 3.3.

3.1. Differential evolution

The DE algorithm is an optimization method based on group evolution process and it computed the optimal solution by three major manipulations: the differential variation process, the binary mutation operation and greedy choose. The major computing process is as follows [20].

At first, the DE algorithm will generate N_p population, which the dimension is D, the solution of the individual is $X_{i,G} = (x_{i1,G}, x_{i2,G}, \ldots, x_{iD,G})$. G stands for the number of iteration, then, generates corresponding variation vector $V_{i,G}$ using different differential variation strategy towards every solution vector $X_{i,G}$. The differential variation strategy used in this paper is DE/rand/2 [21]:

$$V_{i,G} = X_{r_1^i,G} + F \cdot (X_{r_2^i,G} - X_{r_3^i,G}) + F \cdot (X_{r_4^i,G} - X_{r_5^i,G}). \quad (15)$$

After the operation, to generate the final probing solution $U_{i,G}$ based on every solution vector $X_{i,G}$ and variation vector $V_{i,G}$ using the binary mutation operation:

$$u_{ij,G} = \begin{cases} v_{ij,G} & rand() \le C_R \quad or \quad j = j_{rand} \\ x_{ij,G} & rand() > C_R \quad and \quad j \ne j_{rand}. \end{cases} \quad (16)$$

In Equation (16), $u_{i,G}$, $v_{i,G}$ and $x_{i,G}$ are the jth vector of the final probing solution $U_{i,G}$, variation vector $V_{i,G}$ and solution vector $X_{i,G}$ respectively. C_R stands for the mutation probability, j_{rand} is the stochastic number.

Finally, we conduct the choosing operator operation between probing solution $U_{i,G}$ and solution vector $X_{i,G}$, and then the best solution will be regard as the new solution $X_{i,G+1}$ and will be stored in the next generation.

3.2. MPGWO algorithm

The GWO algorithm imitates the leadership hierarchy and hunting mechanism of grey wolves in nature proposed by Mirjalili et al. [14]. Grey wolves are considered to be at the top of food chain and they prefer to live in a pack. Four types of grey wolves such as alpha (α), beta (β), delta (δ) and omega (ω) are employed for simulating the leadership hierarchy. In order to mathematically model the social hierarchy of wolves while designing GWO, we consider the fittest solution as the alpha (α). Consequently, the second and third best solutions are named as beta (β) and delta (δ), respectively. The rest of the candidate solutions are assumed to be omega (ω). Figure 1 shows three main steps of GWO algorithm, namely hunting, chasing and tracking for prey, encircling prey and attacking prey which are implemented to design GWO for performing optimization.

Recently, a multi-population version of the GWO (MPGWO) was proposed which extends the idea of the original GWO for solving optimization problems with multiple and conflicting populations. In MPGWO, a fixed sized external archive is integrated to the GWO for saving and retrieving the Pareto optimal solutions. This archive is then employed to define the social hierarchy and simulate the hunting behaviour of grey wolves in multi-objective search spaces, and share and exchange information among different populations in order to improve the diversity of the population for the purpose of optimal solutions. The pseudo code of the MPGWO algorithm is taken as in Table 1.

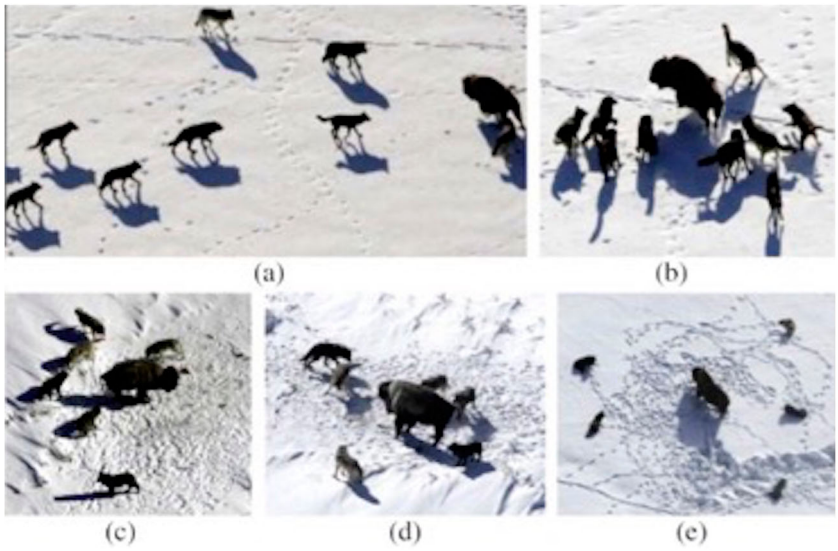

Figure 1. Hunting behaviour of grey wolves: (a)–(c) chasing and tracking prey; (d) encircling prey and (e) attacking prey.

Table 1. The pseudo code of multi-population grey wolf optimization algorithm.

The pseudo code of MPGWO Algorithm
Step 1 Parameter Initialization: The maximum number of iterations, the size of each population and the searching space of the corresponding population *Max iteration = The maximum number of iterations;* *Agent number = The size of each population;* *Dim-1 = the dimension of the searching space;* *For i = 1:3 (i is the number of the population);* *Ub(i) = the searching upper limit of the ith population;* *Lb(i) = the searching lower limit of the ith population;* *End* **Step 2 Population Initialization**: initial each population respectively after the parameter initialization process. *J* is the serial number of the grey wolf in the populations *For i = 1:3* *Position(j,:) = rand(agent number,1). *(Ub(i)- Lb(i))+ Lb(i)* *End* **Step 3 Computing the Fitness**: Each population was coded and to calculate the fitness of each wolf. *For i = 1:3* *For j = agent number* *C = Position (j, 1);* *γ = Position (j, 2);* *Fitness(C) = Function (C,γ);* *End* *End* **Step 4 Preserving the position of grey wolves which fitness ranking in the top three in each population**: Preserve the fitness value and position of grey wolves which fitness ranking in the top three in each population, set the wolf as the heading wolf with the greatest fitness value, and ranking the fitness value of the heading wolf in three populations. Then set the three heading wolves as the ones with the greatest fitness value. Other individuals in the population close to the new heading wolf. *For i = 1:3* *Alpha_score(i) = fitness(index(1));* *End* **Step 5 Iterative Optimization**

3.3. The proposed hybrid DE-MPGWO algorithm

The new proposed hybrid DE-MPGWO algorithm using the DE and MPGWO as the evolution method has the ability of meme evolution derived from Frog Leaping algorithm (FLA) in order to improve the performance when taking the advantage of the two algorithms. The detailed implementation of the proposed optimization algorithm is described in Table 2.

Table 2. The pseudo code of DE-MPGWO algorithm.

The pseudo code of DE-MPGWO algorithm
Step 1: In the solution space, generate *ND* dimensional solutions as initial populations randomly, the total number of evolution iterations is $l_{iter\,\max}$, the number of iterations belong to each subpopulation isl_{iter}, $C_n = \frac{l_{iter\,\max}}{l_{iter}}$. **Step 2:** The population is divided into N_k subpopulations randomly. **Step 3:** Choose k subpopulations randomly in the N_k subpopulations while $1 < k < N_k$, using the DE algorithm to compute l_{iter} generations in the iteration process respectively. Regarding to the rest $N_k - k$ subpopulations, we divide them into three grey wolf populations. Using the MPGWO algorithm to compute l_{iter} generations in the corresponding iteration process respectively. In the entire iteration process, recording all the changes of the optimal value to all the populations. **Step 4:** Mixing N_k subpopulations to obtain the new population, judging whether the number of the iteration of iteration of local search reaches the designated number C_n, if so, the iteration stops, if not, turn to Step 2; **Step 5:** The algorithm is termination.

4. The proposed HI-DKIELM

In this part, the traditional extreme learning machine (ELM) is to be extended to HI-DKIELM based on kernel incremental extreme learning machine and deep learning network. The proposed HI-DKIELM consists of an input layer, output layer and some hidden layer of cascade. The structure of the HI-DKIELM is shown in Figure 2. In the training process, we utilized the DE-MPGWO optimization algorithm while given in this paper to optimize the output weight for the purpose of robustness. The initial data after the subtract through k hidden layer is to obtain the input feature X^k, then mapping the input feature using the kernel function. The detailed implementation process of the proposed HI-DKIELM is given in Table 3.

In this paper, the proposed hybrid intelligent HI-DKIELM extracts the input data layer by layer in order to obtain more effective features, which are conducive

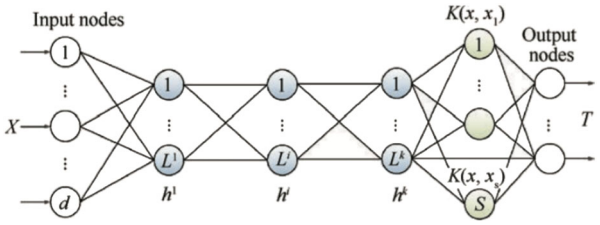

Figure 2. The structure of proposed DKI-ELM.

Table 3. The pseudo code of HI-DKIELM algorithm.

The pseudo code of HI-DKIELM algorithm

Step 1: Suppose training samples, $x_i = [x_{i1}, \quad x_{i2}, \quad \cdots \quad x_{in}]^T$, $t_i \in R$, set the expected network output error function is η, the prediction error of the output is $\varphi(x_i, t_i)$, the number of the hidden node $L = 0$, network error $e_0^* = T$, the number of iterations is $k = 0$;
Step 2: Set the hidden layer nodes$L = L + \lambda$., when $\lambda = 1$,indicate that we add one node in the hidden layer;
Step 3: Computing the prediction error**:**

$$E_R = \frac{1}{2N}\sum_{i=1}^{N}|f_i(x) - f|$$

Step 4:Y_{L^*} Computing the Optimal parameters of the hidden layer node Y_{L^*} based on the DE-MPGWO algorithm which proposed in this paper, then computing the output weigh

$$\beta_{L^*}\text{:}\beta_{L^*} = \frac{E_L \cdot H_L^T(Y_{L^*})}{H(Y_{L^*}) \cdot H_L^T(Y_{L^*})}$$

Step 5: Computing the output error:
$E_L = E_{L-1} - \beta_{L^*}\|H_L(Y_{L^*}), x\|$
if $E_L < \eta, E_R < \varphi$, the training step termination, else, turn to Step2;
Step 6: Suppose $A = \left[\frac{1}{C} + K_{ELM}\right]$, for the t moment is A_t, for the $t+1$ moment is A_{t+1}, computing the generalized inverse of A_{t+1};
Step 7: Update the data online, computing the output $\hat{Y}_{test}$;
Step 8: The algorithm is termination.

to distinguish confused types easily and improving classification accuracy. In addition, these abstract features are not original input features, but the kernel function calculation could instead of the inner product calculation in the high-dimensional space, which is conducive to further improving the accuracy of classification.

5. Result and discussion

5.1. Experimental settings

In this section, we will provide a wide range of different experimental results in different quarters to access the effectiveness of the proposed new method.

In the experiment, the system operating environment is Intel (R) Xeon (R) CPU E3-1231 v3@ 3.40 GHz 3.40 GHz, memory 16 GB, running Win7 PC, and the programming language is Matlab2013a. In order to verify the validity and robustness of the proposed HI-DKIELM algorithm, the experiments are including four parts:

(1) At first, we test the performance and robustness of the proposed DE-MPGWO optimization algorithm, which proposed in Section 3 while using for obtaining the parameter of the hidden layer node and output weight.

Table 4. UHI real data set.

Data set	Training samples	Test samples	Number of features	Purpose
Firedman	18,000	12,000	11	Regression
CCPP	7000	4300	4	Regression
Servo	100	67	6	Regression
California Housing	4400	2000	8	Regression
CCS	1000	900	9	Regression
Abalone	4100	3000	8	Classification
Energy Efficiency	768	500	8	Classification
Boston Housing	466	300	13	Classification
Bank	3000	1300	8	Classification
Delta Ailerons	2800	1700	5	Classification

(2) For the HI-DKIELM while proposed in Section 4, the number of hidden layers has an important impact for the performance of neural networks. Based on the Abalone database, we test the impact of different number of hidden layers for the network structure.
(3) In order to test the generalization performance of the proposed HI-DKIELM algorithm using 10 groups data of UHI real data set, we compare it with the common CI-ELM, EI-ELM, ECI-ELM and DCI-KELM for regression problems.
(4) In order to test the generalization performance of the proposed HI-DKIELM algorithm using 10 groups data of UHI real data set, we compare it with the common CI-ELM, EI-ELM, ECI-ELM and DCI-KELM for classification problems.

5.2. Evaluate the performance and robustness of the DE-MPGWO optimization algorithm

In this section, we will evaluate the performance and robustness of the DE-MPGWO optimization algorithm proposed in Section 3.

In this experiment, we are using 10 typical functions, which are stated in Table 5 in order to check the optimization ability. The dimension of the solution in every typical function D is set as 30, the range of the solution of the typical function F_{n6} is set as $[-100, 100]$, F_{n9} is set as $[-500, 500]$, the remains set as $[-30, 30]$.

In order to compare the performance of the DE-MPGWO algorithm, we compare it with three basic optimization algorithms: DE, PSO and FLA and DEPSO that proposed in reference [24] while it is our former work. The parameters in each method are given in Table 6.

In the numerical experiment, the scales of the population in four algorithms are all the same that means $N_p = 40$, the number of the iteration in each method is 2000. For each typical function, the times of the optimization using four optimization algorithm is 50 and the average optimization value will be utilized as the final optimization result. Table 7 outlined the

Table 5. Typical optimization functions.

Function Name	Equations
Sphere	$F_{n1} = \sum_{i=1}^{D} x_i^2$
Quadric	$F_{n2} = \left(\sum_{i=1}^{D} x_i^2\right)^2$
Griewank	$F_{n3} = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley	$F_{n4} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp$ $(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i) + 20 + e$
Rosenbrock	$F_{n5} = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$
Rastrigin	$F_{n6} = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
Schwefel's 1.2	$F_{n7} = \left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2\right)(1 + 0.4\lvert N(0,1)\rvert)$
Griewank with noise	$F_{n8} = \sum_{i=1}^{D} \frac{(x_i - N(0,1))^2}{4000}$
Schwefel	$F_{n9} = 418.9828D - \sum_{i=1}^{D} x_i \sin(\lvert x_i\rvert^{1/2})$
Schwefel's 2.22	$F_{n10} = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$

Table 6. The parameters in each method.

Method	Parameter
DE	$F = 0.5, C_R = 0.3$
PSO	$w = 0.729, c_1 = c_2 = 1.495$
FLA	$C = 1.2$
DEPSO	$N_k = 4, I_{iter} = 10, k = 4$
DE-MPGWO	$N_k = 4, I_{iter} = 10, k = 4$

optimization result using DE, PSO, FLA, DEPSO and DE + MPGWO methods while making use of 10 typical functions.

From the optimization results given in Table 7, we can found that when we take optimization experiment to the typical function F_{n2}, the searching performance obtained using four optimization algorithms can reach an ideal result. While the optimization result taking another nine typical functions can be summarized as follows: (1) For the precision of searching results, the proposed DE-MPGWO optimization method is better than the DE, PSO and the FLA methods obviously when taking to the other nine typical functions, and it can obtain more precise solution. (2) For the ability of out of local minimum, the PSO algorithm falls into the minimum value point quickly and the length which play the major role in the time domain in the optimization periods is very short; for the proposed DE-MPGWO strategy, its can out of the local minimum continuously in the iteration process in order to search the optimal solution and it have a better searching ability. In conclusion, the proposed DE-MPGWO optimization method has a better improvement in the searching optimization ability and has a good balance to the searching optimization precision and convergence speed.

5.3. Setting the parameters of HI-DKIELM

In HI-DKIELM, the number of different network hidden layers has an important impact on the performance of neural networks. Based on the Abalone database, it assumes that the number of hidden layers is 1–6 respectively, and the number of nodes in each layer is 20. In this experiment, the impact of different hidden layer numbers on the network structure is tested. Each network structure is tested 10 times and the experimental results are as shown in Figure 3.

From the results shown in Figure 3, it can be seen that the testing accuracy does not increase with the increase of the hidden layer data. When the number of hidden layers is 3, the performance of the proposed method is the most stable. When the number of hidden layers continues to increase, the testing accuracy decreased, so in our following experiment, the number of hidden layers was chosen to be 3.

In HI-DKIELM, the kernel function parameters γ and the regularization parameters C have a great influence on the performance. At present, most of the existing methods are selected by the cross-validation method. In this paper, the values of the two parameters are changed from the range $10^0 \sim 10^{10}$ at the same time, and the testing accuracy is calculated. The test accuracy and the values of the two parameters are plotted as a surface, shown in Figure 4.

It can be seen from the experiment results shown in Figure 4 that when the regularization parameter C takes a small value, the performance is poor and varies violently with the kernel function. When the regularization parameter increases, the performance tends to be stable and has the highest accuracy simultaneously.

5.4. Evaluate the performance of the proposed HI-DKIELM based on the regression problem

In this part, we will evaluate the performance of the proposed HI-DKIELM based on the regression problem. In the comparing experiment, the purpose is to evaluate the generalization and robustness of the HI-DKIELM, so we compare it with four basic ELM algorithms: CI-ELM, EI-ELM, ECI-ELM and DCI-KELM. In the experiment, the number of initial hidden layer neurons in the neural network is one and the number of hidden layer neurons in each iteration is increased by one, all the extreme learning machines have the same hidden layer neurons and the same number of iterations. The comparison of the training error and the testing error on the regression problem test is outlined

Table 7. Typical optimization function optimization result.

Function	Performance	DE	MPGWO	FLA	DEPSO	DE-MPGWO
F_{n1}	Mean	1.7667e+002	1.2494e+003	1.1951e+002	1.2008e-009	0.9854e-009
	Variance	7.3438e+001	2.5787e+002	4.1270e+001	7.4548e-010	5.8208e-010
F_{n2}	Mean	0.0000e+000	1.2494e+000	0.0000e+000	4.3528e-007	2.0305e-007
	Variance	0.0000e+000	1.9059e-002	0.0000e+000	8.2034e-007	6.0356e-007
F_{n3}	Mean	9.8590e-001	1.2928e+000	1.0011e+000	8.4011e-004	4.0517e-004
	Variance	9.2600e-002	6.0500e-002	3.5900e-002	1.5000e-003	1.0714e-003
F_{n4}	Mean	9.1589e+000	1.6711e+001	7.8190e-000	8.6480e-004	7.0882e-004
	Variance	1.4730e+000	7.4780e-001	1.1279e-000	7.0593e-006	4.4363e-006
F_{n5}	Mean	2.9764e+005	1.3255e+007	1.2866e+005	2.6678e+001	1.2678e+001
	Variance	1.9895e+00	6.2356e+006	7.4297e+004	9.3772e+000	7.8358e+000
F_{n6}	Mean	3.6770e+002	1.6137e+003	8.1555e+002	1.4644e+002	1.1108e+002
	Variance	7.3999e+001	3.3505e+00	1.2285e+001	1.2235e+001	0.5141e+001
F_{n7}	Mean	2.3981e+004	3.7054e+005	7.7842e+002	2.3319e+002	1.4922e+002
	Variance	9.3795e+003	9.5826e+004	3.7931e+002	1.4052e+001	0.8839e+001
F_{n8}	Mean	3.8400e-002	3.1190e-001	1.4609e-003	8.9489e-004	6.1414e-004
	Variance	1.3400e-002	6.6700e-002	5.2516e-002	1.3616e-004	0..4517e-004
F_{n9}	Mean	6.8456e+003	7.4513e+003	7.5097e+003	4.4974e+003	2.2663e+003
	Variance	1.4465e+003	6.2601e+002	6.1316e+002	3.6771e+002	1.6178e+002
F_{n10}	Mean	5.5864e+001	2.0366e+002	8.8238e+002	1.0276e-004	0.8385e-004
	Variance	1.2322e+001	2.3497e+001	4.0068e+001	3.7921e-005	2.2097e-005

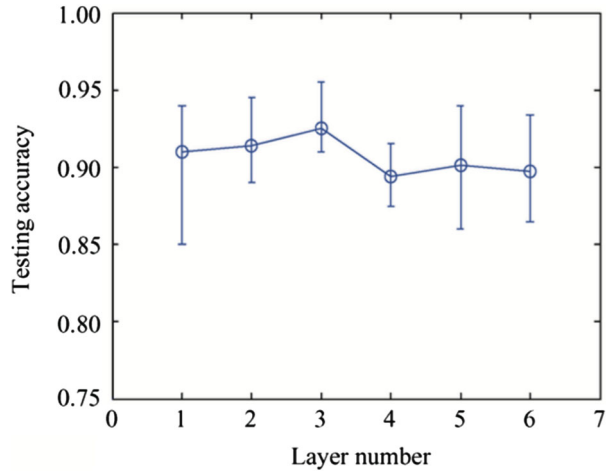

Figure 3. The comparison of the testing accuracy under different hidden layer nodes.

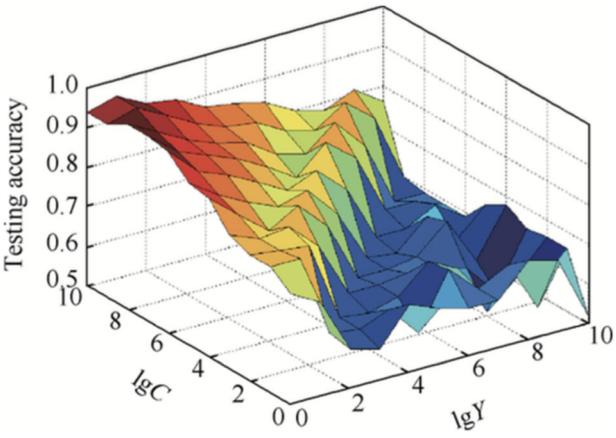

Figure 4. The comparison of the testing accuracy under different parameters.

in Table 8. Table 9 is the comparison of the network complexity and training time on the regression problem. The value in the bracket is the RMSE which is the error termination condition.

From the results shown in Table 8, we can find that for the regression problem, the accuracy of the HI-DKIEM method proposed in this paper has improved significantly compared with the other four ELM algorithms. For example, taking account to the CCPP database, when the maximum hidden layer node number is 100 and the error termination condition criteria RMSE is 0.052, the training error and the testing error of the HI-DKIELM are 0.0417 and 0.0435 respectively. However, for the DCI-KELM method, the training error is 0.0513 and the testing error is 0.0508, for the ECI-KELM algorithm, the training error and the testing error are 0.0535 and 0.0604 respectively. So from the perspective of training error and testing error, the generalization and robustness of the HI-DKIELM is better than other four ELM methods obviously.

The comparison of network complexity and training time for the regression problem between the five ELM is given in Table 9. From the experiment results, we can find that the network complexity and training time of the HI-DKIEM method proposed in this paper has improved significantly compared with the other four ELM algorithms. For example, taking account the CCPP database, when the termination condition criteria RMSE is 0.052, the network nodes and training time of the HI-DKIELM are 27.06 and 1.0277 s respectively. However, for the DCI-KELM method, the network nodes are 45.35 and the training time is 2.0743S, for the ECI-KELM algorithm, the network nodes and training time are 111.92 and 3.0178 s respectively. So from the perspective of network nodes and training time, the generalization and robustness of the HI-DKIELM is better than other four ELM methods obviously.

5.5. Evaluate the performance of the proposed HI-DKIELM based on the classification problem

In this part, we will evaluate the generalization performance of the proposed HI-DKIELM based on the

Table 8. Comparison of training error and testing error on regression problem.

		CI-ELM		EI-ELM		ECI-ELM		DCI-KELM		HI-DKIELM	
Data set	Node	Training error	Testing error	Training error	Testing error	Training error	Testing error	Training error	Testing error	Training error	Testing error
Firedman (0.1)	50	0.1061	0.1672	0.1465	0.1589	0.1424	0.1353	0.0956	0.1091	**0.0905**	**0.0951**
CCPP (0.052)	100	0.0576	0.0616	0.0551	0.0529	0.0535	0.0604	0.0513	0.0508	**0.0417**	**0.0435**
Servo (0.155)	50	0.1677	0.1784	0.1612	0.1601	0.1547	0.1542	0.1531	0.1514	**0.1229**	**0.1219**
California Housing (0.15)	100	0.1623	0.1588	0.1486	0.1533	0.1521	0.1509	0.1467	0.1403	**0.1189**	**0.1154**
CCS (0.045)	100	0.0671	0.0727	0.0529	0.0513	0.0447	0.0424	0.0431	0.0382	**0.0346**	**0.0318**

Table 9. Comparison of the network complexity and training time comparison on regression problem.

	CI-ELM		EI-ELM		ECI-ELM		DCI-KELM		HI-DKIELM	
Data set	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)
Firedman (0.1)	140.26	2.2183	87.82	4.2162	62.37	4.0021	19.75	3.1134	**13.14**	**1.4899**
CCPP (0.052)	390.76	2.0104	305.18	2.5622	111.92	3.0178	45.35	2.0743	**27.06**	**1.0277**
Servo (0.155)	70.01	0.0381	58.56	0.4233	14.22	0.4511	11.74	0.0373	**5.53**	**0.0281**
California Housing (0.15)	220.73	0.0782	19.24	1.0515	112.08	1.4203	17.42	0.7216	**14.35**	**0.3945**
CCS (0.045)	70.22	0.5329	65.02	0.7439	51.23	0.8241	49.33	0.4125	**22.38**	**0.2267**

classification problem. In the comparing experiment, the purpose is to evaluate the generalization and robustness of the HI-DKIELM, so we compare it with four basic ELM algorithms: CI-ELM, EI-ELM, ECI-ELM and DCI-KELM. In the experiment, the number of initial hidden layer neurons in the neural network is one and the number of hidden layer neurons in each iteration is increased by one, all the extreme learning machines have the same hidden layer neurons and the same number of iterations. The comparison of the training error and the testing error on the classification problem test are outlined in Table 10. Table 11 is the comparison of the network complexity and training time on the classification problem.

From the results shown in Table 10, we can find that for the classification problem, the accuracy of the HI-DKIEM method proposed in this paper has improved significantly compared with the other four ELM algorithms. For example, taking account to the Boston Housing database, when the maximum hidden layer node number is 100 and the error termination condition criteria RMSE is 0.1, the mean value and the standard deviation of the HI-DKIELM are 98.21and 0.0038 respectively. However, for the DCI-KELM method, the mean value is 93.01 and the standard deviation is 0.0041, for the ECI-KELM algorithm, the mean value and the standard deviation are 84.82 and 0.0072 respectively. So from the perspective of the mean value and the standard deviation, the generalization and robustness of the HI-DKIELM are better than other four ELM methods obviously.

The comparison of network complexity and training time for the regression problem between the five ELM is given in Table 11. From the experimental results, we can find that the network complexity and training time of the HI-DKIEM method proposed in this paper have improved significantly compared with the other four ELM algorithms. For example, taking account to the CCPP database, when the termination condition criteria RMSE is 0.1, the network nodes and training time of the HI-DKIELM are 19.42 and 0.0774 s respectively. However, for the DCI-KELM method, the network nodes are 22.06 and the training time is 0.0942 s, for the ECI-KELM algorithm, the network

Table 10. Comparison of training error and testing error on classification problem.

		CI-ELM		EI-ELM		ECI-ELM		DCI-KELM		HI-DKIELM	
Date set	Node	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
Abalone (0.075)	50	83.18	0.0022	84.02	0.0018	93.76	0.0014	94.23	0.0015	**103.48**	**0.0010**
Energy efficiency (0.045)	100	90.18	0.0014	92.42	0.0016	94.12	0.0011	96.52	0.0008	**104.97**	**0.0006**
Boston Housing (0.1)	50	68.02	0.0113	71.98	0.0101	84.82	0.0072	93.01	0.0041	**98.21**	**0.0038**
Bank (0.065)	100	90.21	0.0217	92.03	0.0122	91.82	0.0124	95.89	0.0076	**103.44**	**0.0071**
Delta Ailerons (0.04)	100	89.23	0.0372	86.89	0.0221	92.23	0.0117	92.71	0.0108	**101.74**	**0.0083**

Table 11. Comparison of the network complexity and training time comparison on classification problem.

	CI-ELM		EI-ELM		ECI-ELM		DCI-KELM		HI-DKIELM	
Data set	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)
Abalone (0.075)	143.05	0.2852	113.94	0.9216	23.64	1.1215	19.15	0.6211	**17.11**	**0.6159**
Energy efficiency (0.045)	72.21	0.1502	80.43	0.3023	67.03	0.2822	50.21	0.2023	**46.89**	**0.1756**
Boston Housing (0.1)	38.41	0.0973	21.11	0.0734	39.17	0.1168	22.06	0.0942	**19.42**	**0.0774**
Bank (0.065)	199.43	0.6745	170.27	0.7891	10.21	0.8922	10.97	0.6276	**8.47**	**0.5508**
Delta Ailerons (0.04)	332.69	1.1239	290.21	2.1742	39.66	1.3653	36.03	0.6912	**30.26**	**0.6372**

nodes and training time are 39.17 and 0.1168 s respectively. So from the perspective of network nodes and training time, the generalization and robustness of the HI-DKIELM is better than other four ELM methods obviously.

6. Conclusion

In this paper, a novel HI-DKIELM based on KIELM and a new DE-MPGWO method under the deep learning network structure is proposed in this paper. The proposed HI-DKIELM can reduce the redundant network nodes due to the reason of ineffective iteration increase and lower learning efficiency.

The main contribution of this paper can be summarized as follows: (1) An HI-DKIELM network classifier is designed. HI-DKIELM consists of a deep learning network and kernel incremental extreme learning machine of cascade, where the input data through the deep leaning network to extract more information and boost the separability can achieve higher dimensional spatial mapping, then the ELM network can be utilized to provide a superior classification surface. In this way, the HI-DKIELM proposed in this paper combines the advantages of the deep learning network and the KIELM network and can improve the performance effectively. (2) In order to explore an optimal parameter belonging to the Extreme Learning Machine, an appropriate hybrid intelligent optimize algorithm for HI-DKIELM is presented. The proposed hybrid DE-MPGWO algorithm optimizes the method using the global search ability of the DE algorithm and the local search capability of multi-group grey wolf algorithm.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work was supported by the National Natural Science Foundation of China under Grant 61841103, by the National Natural Science Foundation of Hunan Province of China under Grant 18JJ4039, by the Key Project of Education Department of Hunan Province of China under Grant 15A044, 16K024, 17A048.

References

[1] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing. 2006;70(1):489–501.
[2] Arriaga RI, et al. Visual categorization with random projection. Neuralcomputing. 2015;27(10):2132–2147.
[3] Dasgupa S, Stevens CF. A neural algorithm for a fundamental computing problem. Science. 2017;358(6364):793–796.
[4] Huang GB, Li MB, Chen L, et al. Incremental extreme learning machine with fully complex hidden nodes. Neurocomputing. 2008;71:576–583.
[5] Huang GB, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71:3460–3468.
[6] Huang GB, Chen L. Convex incremental extreme learning machine. Neurocomputing. 2007;70:3056–3062.
[7] Wang W, Zhang R. Improved convex incremental extreme learning machine based on enhanced random search. Electric Eng Electron Eng. 2014;238:2033–2040.
[8] Zhang R, Lan Y, Huang G B, et al. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Trans Neural Networks Learning Sys. 2012;23:365–371.
[9] Guo L, Hao J H, Liu M. An incremental extreme learning machine for online sequential learning problems. Neurocomputing. 2014;128:50–58.
[10] Yang YM, Wang YN, Yuan XF, et al. Hybrid chaos optimization algorithm with artificial emotion. Appl Math Comput. 2012;218(11):6585–6591.
[11] Yang YM, Wang YN, Yuan XF. Parallel chaos search based incremental extreme learning machine. Neural Process Lett. 2013;37:277–301.
[12] Zhu QY, Qin AK. Evolutionary extreme learning machine. Pattern Recognit. 2005;38(10):1759–1763.
[13] Han F, Yao HF, Ling QH. An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing. 2013;116:87–93.
[14] Miche Y, Sorjamaa A, Bas P, et al. OP-ELM: Optimally pruned extreme learning machine. IEEE Trans Neural Networks. 2010;21(1):158–162.
[15] Lin MJ, Luo F, Su CH, et al. An improved hybrid intelligent extreme learning machine. Control Decis. 2015;30(6):1078–1084.
[16] Zhang WB, Ji HB, Wang L, et al. Multiple hidden layer output matrices extreme learning machine. Syst Eng Electron. 2014;36(8):1656–1660.
[17] Pang S, Yang X-Y, Zhang Y, et al. Application of deep kernel extreme learning Machine in aero engine components fault diagnosis. J Propul Technol. 2017;37(11):2613–2620.
[18] Jones AJ. New tools in non-linear modeling and prediction. Comput Manage Sci. 2004;1(2):109–149.
[19] Storn R, Price K. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim. 1997;11(4):341–359.
[20] Qin AK, Huang AL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput. 2009;13(2):398–417.
[21] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw. 2014;69(3):46–61.
[22] Sulaiman MH, Mustaffa Z, Mohamed MR, et al. Using the gray wolf optimizer for solving optimal reactive power dispatch problem. Appl Soft Comput. 2015;32(C):286–292.
[23] Song X, Tang L, Zhao S, et al. Grey wolf optimizer for parameter estimation in surface waves. Soil Dynam Earthquake Eng. 2015;75(8):147–157.
[24] Di W, Qin W. Multimodal biometrics fusion based on TER and hybrid intelligent multiple hidden layer probabilistic extreme learning machine. Int J Comput Intelligent Sys. 2018;11(1):936–950.