A hybrid deep kernel incremental extreme learning machine based on improved coyote and beetle swarm optimization methods

Di Wu^{1,2} · Ting Li¹ · Qin Wan¹

Received: 7 March 2021 / Accepted: 24 July 2021 / Published online: 14 August 2021
© The Author(s) 2021

Abstract

The iteration times and learning efficiency of kernel incremental extreme learning machines are always affected by the redundant nodes. A hybrid deep kernel incremental extreme learning machine (DKIELM) based on the improved coyote and beetle swarm optimization methods was proposed in this paper. A hybrid intelligent optimization algorithm based on the improved coyote optimization algorithm (ICOA) and improved beetle swarm optimization algorithm (IBSOA) was proposed to optimize the parameters and determine the number of effectively hidden layer neurons for the proposed DKIELM. A Gaussian global best-growing operator was adopted to replace the original growing operator in the intelligent optimization algorithm to improve COA searching efficiency and convergence. In the meantime, IBSOA was designed based on tent mapping inverse learning and dynamic mutation strategies to avoid falling into a local optimum. The experimental results demonstrated the feasibility and effectiveness of the proposed DKIELM with encouraging performances compared with other ELMs.

Keywords Kernel incremental extreme learning machine · Coyote optimization algorithm · Beetle swarm optimization algorithm · Hybrid intelligence

Introduction

Since proposed by Huang et al., extreme learning machine (ELM) has shown fast training speed and incomparable classification ability in the fields of machine learning and neural networks during the past decades [1, 2]. From the perspective of network structure, ELMs are single-hidden layer feedforward networks (SLFNs). ELMs are an efficient solution for SLFNs to address drawbacks such as learning rate, training epochs, and local minima [3] as they do not require any iterative techniques [4, 5].

The hidden nodes, input weights, and hidden bias of ELM are all generated randomly. Meanwhile, the output weights of neurons are calculated using the conventional Moore–Penrose inverse based on the least square method. Therefore, the parameter tuning process is not required to generate the network model before training, which is entirely irrelevant to the training data. ELM has incomparable benefits compared to the other machine learning methods in terms of training speed, global optimality, and generalization performance.

Although the effectiveness of ELM has been proved in some specific applications, its training data are added one by one or chunk by chunk. Therefore, a wide range of improved ELMs was developed. The incremental ELM (I-ELM) proposed by Huang et al. [6] could calculate the output weights for the ever-increasing hidden nodes more accurately, but it remained a universal approximate approach. The adaptive growth ELM (AG-ELM) proposed by Zhang et al. [7], which consisted of the hidden nodes, the incremental renewal of the network weights, and the sequential generation of the grouping new networks, could approximate the nonlinearity functions effectively. ELM can also be transformed into an online version based on the online sequence analysis [8]. As for generalization, the kernel ELM (KELM) based on the

✉ Di Wu
wudi6152007@163.com

Ting Li
258960781@qq.com

Qin Wan
15798021@qq.com

¹ College of Electrical and Information Engineering, Hunan Institute of Engineering, Xiangtan 411104, China

² Hunan Province Cooperative Innovation Center for Wind Power Equipment and Energy Conversion, Hunan Institute of Engineering, Xiangtan 411104, China

kernel function was proposed in [9] for classification problems. A sequential online ELM based on KELM is proposed in [10] for online learning. Through optimizing the input weights, Sun et al. [11] proposed a differential evolutionary ELM under the compact network structure construction to improve the generalization performance. However, the current version of I-ELM still faces redundant nodes that continually increase the network structure complexity and reduce the learning efficiency. The convergence rate and the prediction ability are still low due to the unfavorable number of hidden layer nodes and the sensitiveness to the new data.

Training speed and learning accuracy crucial for ELM depend heavily on the parameter optimization process. Therefore, researchers proposed several intelligent optimization methods to compute ELM parameters more precisely. Improving the training speed and learning accuracy of ELMs based on bionics methods is becoming a research focus. A differential evolution algorithm is utilized in [12] to optimize the ELM input parameters. An adaptive differential evolution algorithm is given in [13] to optimize the hidden layer node parameters. The MP generalized inverse method was utilized to calculate the output weights. An improved particle swarm optimization algorithm is proposed in [14] to optimize the hidden layer node parameters.

Although the traditional differential evolution optimization method has strong global optimization ability, it could not avoid the premature problem. Traditional particle swarm optimization algorithm has low searching speed despite its local optimization merits. Therefore, a hybrid optimization approach called DEPSO based on differential evolution algorithm and the particle swarm optimization method is given in [15] to optimize the hidden layer nodes. A novel hybrid optimization method given in [16] took advantage of the global search ability of the differential evolution algorithm and the local search capability of the multi-group gray wolf optimization algorithm. However, there are still problems of local optimum and poor population diversity in later iterations.

As the extension of the research in [16], a Hybrid Intelligent Deep Kernel Incremental Extreme Learning Machine (HI-DKIELM) based on the improved coyote optimization algorithm (ICOA) and improved beetle swarm optimization algorithm (IBSOA) is proposed in this paper. First, COA was improved based on the Gaussian global best-growing operator to enhance the searching efficiency and convergence speed. Second, the tent mapping inverse learning and dynamic mutation strategies were adopted to prevent BSOA from local optimum and modify the population diversity in later iterations. Finally, the hybrid intelligent optimization method was adopted to modify the parameters of a deep kernel incremental extreme learning machine (DKIELM) and improve the training speed and classification accuracy.

This paper is the first proposal of the HCOBSO strategy as a new perspective.

The significant contributions of this paper are summarized as follows:

1. An improved Coyote Optimization Algorithm is proposed based on the Gaussian global best-growing operator to improve the searching efficiency and convergence. In this paper, we introduce the Gaussian global best-growing operator to replace the original growing operator in the COA method to enhance the searching efficiency and convergence speed.
2. An improved Beetle Swarm Optimization Method was designed based on tent mapping inverse learning and dynamic mutation strategies to prevent BSOA from falling into a local optimum. Due to the higher solving speed and accuracy, BSOA has been used in signal positioning and data classification successfully. However, the original BSOA has several drawbacks. First, it falls into the local optimum easily when solving complex optimization problems. Second, the computational complexity is very high when updating the parameters. Third, in multi-dimension function optimization, relying on a single individual search alone increases the possibility of the optimization algorithm running into the local optimum. Therefore, BSOA was improved in this paper using tent mapping inverse learning and dynamic mutation strategies.
3. The novel HCOBSO method was proposed for optimizing the parameters of DKIELM. The proposed hybrid intelligent optimization method took advantage of the global search ability of COA and the local search capability of BSOA.

The remainder of this paper is organized as follows: the related works are briefly reviewed in section “[Preliminary](#)”. Section “[The proposed HCOBSO method](#)” presents the HCOBSO algorithm. Section “[The proposed HI-DKIELM](#)” presents details of the DKIELM. The experiment result is presented in section “[Results and discussion](#)”. Section “[Conclusion](#)” concludes our work and outlines our future work.

Preliminary

In this part, the notation of Kernel Incremental extreme learning machine (KI-ELM), the Gaussian global best-growing operator, the tent mapping inverse learning, and the dynamic mutation strategies was provided for the convenience of understanding the proposed ELM algorithm.

Kernel incremental extreme learning machine

Suppose training set $\{x_i, t_i\}_{i=1}^N$ is composed of N training samples, the input x_i has d dimensions, and t_i is the label of the output, then the output of the ELM is [17, 18]

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

The kernel matrix of KI-ELM can be expressed as

$$K_{\text{ELM}} = HH^T = h(x_i) \cdot h(x_j) = K(x_i, x_j); \quad (2)$$

thus, the output function of KI-ELM can be given as

$$f(x) = h(x)\beta = h(x)H^T \left(\frac{I}{C} + HH^T \right)^{-1} T \\ = \begin{bmatrix} K(x, x_1) \\ K(x, x_2) \\ \vdots \\ K(x, x_N) \end{bmatrix} \left(\frac{1}{C} + K_{\text{ELM}} \right)^{-1} T. \quad (3)$$

In Eq. (3), assuming $A = \left[\frac{1}{C} + K_{\text{ELM}} \right]$, then at moment t , there are

$$A_t = \begin{bmatrix} \frac{1}{C} + K(x_1, x_1) & \cdots & \frac{1}{C} + K(x_1, x_N) \\ \vdots & \cdots & \vdots \\ \frac{1}{C} + K(x_N, x_1) & \cdots & \frac{1}{C} + K(x_N, x_N) \end{bmatrix}. \quad (4)$$

Thus, at $t + 1$, there are

$$A_{t+1} = \begin{bmatrix} \frac{1}{C} + K(x_1, x_1) & \cdots & \frac{1}{C} + K(x_1, x_{N+k}) \\ \vdots & \cdots & \vdots \\ \frac{1}{C} + K(x_{N+k}, x_1) & \cdots & \frac{1}{C} + K(x_{N+k}, x_{N+k}) \end{bmatrix}. \quad (5)$$

Suppose

$$U_t = \begin{bmatrix} K(x_1, x_{N+1}) & \cdots & K(x_1, x_{N+k}) \\ \vdots & \cdots & \vdots \\ K(x_N, x_{N+1}) & \cdots & K(x_N, x_{N+k}) \end{bmatrix} \quad (6)$$

$$D_t = \begin{bmatrix} \frac{1}{C} + K(x_{N+1}, x_{N+1}) & \cdots & \frac{1}{C} + K(x_{N+1}, x_{N+k}) \\ \vdots & \cdots & \vdots \\ \frac{1}{C} + K(x_{N+k}, x_{N+1}) & \cdots & \frac{1}{C} + K(x_{N+k}, x_{N+k}) \end{bmatrix}; \quad (7)$$

Then, Eq. (5) can be simplified as

$$A_{t+1} = \begin{bmatrix} A_t & U_t \\ U_t^T & D_t \end{bmatrix}. \quad (8)$$

For the new input data, there are

$$A_{t+1} = \begin{bmatrix} A_t^{-1} + A_t^{-1} U_t C_t^{-1} U - t^T A_t^{-1} & -C_t^{-1} U_t C_t^{-1} \\ -C_t^{-1} U_t^T A_t^{-1} & C_t^{-1} \end{bmatrix}. \quad (9)$$

The output value $\hat{Y}_{\text{test}}$ can be estimated online as

$$\hat{Y}_{\text{test}} = \begin{bmatrix} K(x_{\text{test1}}, x_1) & \cdots & K(x_{\text{test1}}, x_M) \\ \vdots & \cdots & \vdots \\ K(x_{\text{testN}}, x_1) & \cdots & K(x_{\text{testN}}, x_M) \end{bmatrix} \begin{bmatrix} A_M^{-1} y_1 \\ \vdots \\ A_M^{-1} y_M \end{bmatrix}. \quad (10)$$

Gaussian global best-growing operator

The Gaussian global best-growing operator is based on the best optimal strategy and Gaussian distribution and inspired by the global optimal harmony search algorithm. The best optimal strategy uses the state information of the current globally optimal target to improve the mining ability and enhance inner information sharing. Gaussian distribution, or normal distribution, is different from the uniform random distribution [0,1] utilized in COA. It expands the scope of the rope and improves the global roping ability to a certain extent.

Tent mapping inverse learning

Research showed that in swarm intelligent searching, the convergence performance is affected by the initial population. The larger and the more evenly distributed the populations, the faster the algorithm converges to the optimal solution. Using chaos mapping to initialize the population is random, ergodic, and bounded. Therefore, the search efficiency can be improved.

The initial sequence generated by Tent mapping [19, 20] is more uniform than that by logistic mapping. Therefore, tent mapping was applied to BSOA to initialize the population, and inverse learning strategies were applied to optimize the initial population. Through competition, the better individuals can be selected for the next generation of learning. Therefore, the population searching scope was expanded, and the invalid searching operation was reduced. Thus, the speed of the convergence is improved. The mathematical expression is as follows [21]:

$$x_{n+1} = \begin{cases} 2x_n, & 0 \leq x_n \leq 1/2 \\ 2(1 - x_n), & 1/2 \leq x_n \leq 1 \end{cases}. \quad (11)$$

Defining an inverse solution feasible in D -dimensional space as $x = (x_1, x_2, \dots, x_D)$, $x = [a, b]$, then the inverse solution is $x' = (x'_1, x'_2, \dots, x'_D)$, where $x' = a + b - x_i$.

Dynamic mutation strategies

In later iterations of BSOA, the decreasing population diversity reduces the searching capacity. Dynamic mutation strategies are introduced to increase population diversity in later iterations and improve the convergence accuracy, so that

premature can be avoided. Scholars have proposed a variety of variation algorithms, such as the Gaussian variation and Cauchy variation [22, 23]. Compared with the Gaussian operator, the Cauchy operator has longer wings and can generate a larger range of random numbers, offering a greater chance to jump out of the local optimum. In the meantime, the Cauchy variation takes less time to search nearby areas when the peak value is lower.

The proposed HCOBSO method

Output weight β is the most important parameter of an ELM. Therefore, the number of hidden layer nodes L should be estimated in advance. In this chapter, the HCOBSO method based on ICOA and IBSO is proposed. The motivation of the new idea is stated in section “[Kernel incremental extreme learning machine](#)”. The concepts of ICOA and IBSO were explained in detail in sections “[Gaussian global best-growing operator](#)” and “[Tent mapping inverse learning](#)”. The implementation of the proposed hybrid intelligent optimization method was summed up in section “[Dynamic mutation strategies](#)”. Therefore, an understanding of the proposed optimization method can be facilitated.

Motivations

Although COA and BSO are all wolves’ optimization algorithms, they are different in some sense. In terms of searching, COA simulates the growing and dying processes of the coyote, whereas BSO simulates the Hierarchy and hunting patterns of beetle. In terms of guide pattern, COA uses only one best wolf to guide other wolves growing. Therefore, the two methods were improved, respectively, and then fused to compensate for the drawbacks in each of them. As a result, an HCOBSO method was obtained. The proposed hybrid intelligent optimization method took advantage of the global search ability of COA and the local search capability of BSOA.

The improved COA method

The COA method is inspired by the *Canis latrans* species in North America, and is classified as both swarm intelligence and evolutionary heuristic method. COA method with a structure different from that of other optimization method does not focus on the social hierarchy and dominant rules.

In this paper, we introduce the Gaussian global best-growing operator to replace the original growing operator in the COA method to enhance the searching efficiency and

convergence speed. In the following, each step in the proposed COA framework will be explained in detail.

1. *Parameter initialization and random initialization of the coyote pack* First, the global population of coyotes consisting of N_p packs each with N_c coyotes is initialized. The initial social conditions for the C th coyote of the p th pack of the j th dimension inside the search space are set randomly, as follows [24, 25]:

$$\text{soc}_{c,j} = \text{lb}_j + r_j \times (\text{ub}_j - \text{lb}_j), \quad (12)$$

where lb_j and ub_j represent the lower and upper bounds of the j th decision variable, and r_j is a real random number generated inside the range $[0, 1]$ using uniform probability. Then, the social fitness value of the coyote is evaluated as

$$\text{fit}_c = f(\text{soc}_c). \quad (13)$$

2. *Coyote growing inside the pack.* The information of the coyotes is computed with the COA as the cultural tendency cult of the pack

$$\text{cult}_j = \begin{cases} O_{(N_c+1/2,j)} & N_c \text{ is odd} \\ \left(O_{N_c/2,j} + O_{(N_c+1/2,j)} \right) & \text{otherwise} \end{cases} \quad (14)$$

Therefore, the coyotes are under the alpha influence δ_1 and the pack influence δ_2 . δ_1 means a cultural difference from a random coyote of the pack cr_1 to the alpha coyote. δ_2 means a cultural difference from a random coyote of the pack cr_2 to the cultural tendency of the pack. δ_1 and δ_2 are written as

$$\delta_1 = \text{alpha} - \text{soc}_{\text{cr}_1} \quad (15)$$

$$\delta_2 = \text{cult} - \text{soc}_{\text{cr}_2}. \quad (16)$$

The alpha is the best coyote inside the pack and the alpha of the p th pack of the t th, an instant of the time, is defined as

$$\text{alpha}^{p,t} = \left\{ \text{soc}^{p,t} \mid \arg_{c=\{1,2,\dots,N_c\}} \min f(\text{soc}^{p,t}) \right\}. \quad (17)$$

Unlike the original COA method, in this paper, the new social condition of coyote: new_soc_1 is updated using the Gaussian global best-growing operator through the following equations:

$$\delta_3 = \text{GP} - \text{soc}_{\text{cr}_1} \quad (18)$$

$$\text{new_soc}_1 = \text{soc} + r_{n1} \times \delta_3 + r_{n2} \times \delta_2. \quad (19)$$

The GP is the global best coyote in the current condition, rn_1 and rn_2 represent a real random number using Gaussian normal distribution. Compared with the original COA, there is a big difference between the individuals at the beginning of the searching operation. It can enhance the searching ability at the beginning of the searching operation. While in the latter searching, the difference is become small and the searching space is become small. Therefore, the new social condition of coyote can improve the production capacity.

3. *Birth and death of coyote* The birth of new coyote pups is written as a combination of the social conditions of two-parent and environmental influence

$$\text{pup}_j = \begin{cases} \text{soc}_{cr_{1,j}} & r_j < p_s \text{ or } j = j_1 \\ \text{soc}_{cr_{2,j}} & r_j \geq p_s + p_a \text{ or } j = j_2 \\ R_j & \text{otherwise} \end{cases}, \quad (20)$$

wherein r_j is the random number in the range $[0, 1]$ using uniform probability, j_1 and j_2 are two random dimensions of the problem, R_j is a random number inside the decision variable bound of the j th dimension, and p_s and p_a are scatter probability and association probability, respectively. Besides, p_s and p_a can be written as

$$p_s = 1/D \quad (21)$$

$$p_a = (1 - p_s)/2. \quad (22)$$

To keep the population size static, the COA syncs the birth and death of coyotes, as shown in Alg. 1.

4. *Coyote expulsion and admission* Sometimes, the coyote will leave packs and become solitary or join a pack instead and occurs with probability p_e

$$p_e = 0.005 \cdot N_C^2. \quad (23)$$

Alg. 1: Birth and death of the coyotes

```

1: Compute  $\omega$  and  $\varphi$ 
2: if  $\varphi = 1$  then
3: The pup survives, and the only coyote in  $\omega$  dies
4: else if  $\varphi > 1$  then
5: The pup survives, and the oldest coyote in  $\omega$  dies
6: else
7: The pup dies
8: end if

```

The pseudocode of the proposed COA is described in Alg. 2

Alg. 2: The pseudocode of the proposed COA

```

1: Initialize  $N_p$  packs with  $N_c$  coyotes each (Eq. 16).
2: Verify the coyote's adaptation (Eq. 17)
3: While stopping criterion is not achieved
do
4:   for each  $p$  pack do
5:     Define the alpha coyote of the pack (Eq. 21)
6:     Compute the social tendency of the pack (Eq. 18)
7:     for each  $c$  coyotes of the  $p$  pack do
8:       Update the social condition (Eq. 23)
9:       Evaluate the new social condition
10:      Adaptation
11:    end for
12:    Birth and death (Eq. 24 and Alg. 1)
13:  end for
14:  Transition between packs (Eq. 27)
15:  Update the coyotes' ages
16: end while
17: Selected the best-adapted coyote.

```

To evaluate the robustness of the proposed COA algorithm, the Schaffer function is adopted to test its optimization performance, as shown in Fig. 1.

Figure 1 shows that the proposed COA methods can find the global optimal solution effectively. The optimization results are close to the global extreme reference value, indicating that the proposed COA method has a good optimization ability.

The improved BSO method

The solving speed and precision accuracy of BSO are higher than other optimal methods. BSO is used in signal positioning and data classification successfully. However, the original BSO has several drawbacks. At first, it falls into local optimality easily when solving the complex optimal problem. Second, the computational complexity is very high when updating the parameters. Third, sawyer only relies on a single individual search, increasing the possibility of the optimal algorithm into the bureau in the multi-dimension function optimization. Therefore, the BSO optimal method is improved using Tent mapping inverse learning and dynamic mutation.

Fig. 1 The test performance of Schaffer function

For the population initialization, the Tent mapping inverse learning is adopted to initializing the population as follows [26]:

Fig. 2 The convergence results of the proposed BSO method

1. In the searching space, the Tent mapping inverse learning is adopted to generate N positions x_{ij} of the beetle population as the initial population OB;
2. According to the definition of inverse solving, inverse positions x'_{ij} of each beetle position x_{ij} in the initial population OB are generated as the inverse population FB;
3. The fitness value belongs to the $2N$ beetle populations, sorted using the ascending sort with the combination of the population OB and FB. N beetle population was selected while corresponding top N fitness value as the initial population.

In the later iteration of the optimal beetle method, the diversity of the population will become worse. Therefore, the secondary optimization operation is performed to the beetle population using the dynamic mutation

$$\begin{cases} X^*(t) = X(t) + \eta * C(0, 1) \\ \eta = e^{-\lambda \frac{1}{t}} \end{cases} \quad (24)$$

In Eq. (24), η is the mutation weight and $C(0, 1)$ is a random value which generates by the Cauchy operator with the probability parameter 1. The pseudocode of the improved BSO is shown in Alg. 3.

Alg. 3: The pseudocode of the proposed BSO

```

1: Initialize the BSO population  $X$  with Tent map
   reverse learning strategy, initialize the BSO toward
    $d$  with random solutions.
2: Calculate fitness of  $X$ , find global best BSO as
    $zbest$ , find local BSO as  $gbest$ 
3: for  $i=1$  to  $T$  do
4:    $d=d/norm(d)$ 

5:    $\eta = e^{-\lambda \frac{1}{T}}$ 

6:   for  $j=1$  to  $N$  do
7:      $X_i(j) = X(j) + d(j) * l$ 
8:     Calculate fitness of  $X_i$ 
9:      $X_r(j) = X(j) - d(j) * l$ 
10:    Calculate fitness of  $X_r$ 
11:    BSO toward  $d(\square + 1) = \omega * d(t) + C_1 * Levy(\theta) * (gbest(t) - X(t)) + C_2 * Levy(\theta) * (zbest(t) - X(t))$ 
12:    population:  $X(t + 1) = X(t) + k_1 * step * d(t) * sign(f(X_r(t)) - f(X_i(t))) + k_1 * d(t)$ 
13:    mutation:  $X^*(t) = X(t) + \eta * C(0,1)$ 
14:  end for
15: Calculate fitness of new BSO, if new BSO is
   better, update it in the population  $zbest$  and  $gbest$ 
16: end for

```

To demonstrate the effectiveness of the proposed BSO method, we test the convergence using the unimodal function $\min f(x) = \sum_{i=1}^n x_i^2$ and compare it with GA, PSO, DE, and original BSO methods. The test results are shown in Fig. 2. From the experiment results, we can find that the convergence speed and convergence accuracy are improved slightly while compared with other four optimization approaches. The global searching ability and local searching ability will be balanced, and the convergence speed will be accelerated and bring it to jump out the local minima when we bring up the Tent mapping inverse learning and dynamic mutation.

The proposed HCOBSO method

A new hybrid optimization approach called hybrid Coyote Optimization and Beetle Swarm Optimization (HCOBSO) algorithm is proposed with the advantage of improved COA

Table 1 The pseudocode of hybrid Coyote Optimization and Beetle Swarm Optimization methods (HCOBSO) algorithm

The pseudocode of the HCOBSO algorithm

```

Step 1: In the solution space, generate  $N$   $D$  dimensional solutions
   randomly as initial populations, so the total number of
   evolution iterations is  $I_{itermax}$ , the number of iterations
   belongs to each subpopulation is  $I_{iter}$ ,  $C_n = \frac{I_{itermax}}{I_{iter}}$ .

Step 2: The population falls into  $N_k$  subpopulations randomly.

Step 3: Choose  $k$  subpopulations in the  $N_k$  subpopulations
   randomly while  $1 < k < N_k$ , then using the COA
   algorithm to compute  $I_{iter}$  generations in the iteration
   process, respectively. Regarding the rest  $N_k - k$ 
   subpopulations, to compute  $I_{iter}$  generations in the
   corresponding iteration process using the BSO algorithm.
   In the whole iteration process, the changes of the optimal
   value, recording to all the populations.

Step 4: Obtain the new population after mixing  $N_k$  sub-
   populations, judging whether the number of the 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 terminated.

```

Fig. 3 The structure of the proposed DKIELM

and improved BSO optimization method while deriving from Frog Leaping algorithm (FLA) to improve the generalization performance. The detailed pseudocode of the proposed optimization method is given in Table 1.

The proposed HI-DKIELM

The detailed structure of the proposed HI-DKIELM is given. The design idea of the HI-DKIELM is based on KI-ELM and deep learning network, showing that the HI-DKIELM consists of three parts: an input layer, output layer, and some hidden layer of a cascade, as shown in Fig. 3. During the training process, the HCOBSO optimization method is utilized to optimize the output weight to enhance its robustness. According to k hidden layer, the input feature X^k can be obtained after subtracting the initial input data and then mapping it using the kernel function. The detailed implementation process of the proposed HI-DKIELM is taken in Table 2.

Table 2 The pseudocode of the HI-DKIELM algorithm

The pseudocode of the HI-DKIELM algorithm	
Step 1:	Suppose training samples (X_i, t_i) , $X_i = [X_{i1}, X_{i2}, \dots, X_{in}]$, $t_i \in \text{Reset}$ the expected network output error function is η , the prediction error of the output is φ , 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:	Computing the Optimal parameters of the hidden layer node Y_{L^*} based on the HCOBSO algorithm which proposed in this paper, then computing the output weights β_{L^*} :
	$\beta_{L^*} = \frac{E_L \cdot H_L^T(Y_{L^*})}{H(Y_{L^*}) \cdot H_L^T(Y_{L^*})}$
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 $\hat{Y}_{\text{test}}$;
Step 8:	The algorithm is terminated.

Results and discussion

Experimental settings

Several experiments are provided in diverse ways to demonstrate the effectiveness of the proposed HI-DKIELM approach.

The experiment is tested on a PC with Intel Core I7-8700 at 3.40 GHz and 16 GB RAM. The proposed method is implemented using Matlab2013a, and the code for the other models comes directly from the code published by the respective authors. To verify the effectiveness and robustness of the HI-DKIELM algorithm, the experiment is divided into five parts:

1. First of all, in section “[Evaluation of the performance of the HCOBSO optimization algorithm using the CEC2017 and CEC2019 database](#)”, the CEC2017 and CEC 2019 database is adopted to test the performance and robustness of the HCOBSO optimization algorithm proposed in section “[The proposed HCOBSO method](#)” and compared with other incomplete optimization methods to test the contribution of each improvement.

2. Section “[Performance evaluation of the HCOBSO optimization algorithm on typical functions](#)” focuses on the proposed HCOBSO optimization method is tested on five typical optimization functions to check the optimization ability.
3. Section “[Selection of hyperparameters](#)” focuses on selecting hyperparameters in HI-DKIELM, including the penalty factor C and the number of hidden nodes L .
4. Section “[Regression analysis](#)” focuses on the regression problems. The generalization performance of the HI-DKIELM algorithm is tested with 10 sets of real UHI data sets and compared it with traditional ELM, I-ELM, EM-ELM, EI-ELM, and B-ELM.
5. Section “[Classification analysis](#)” focuses on the classification problems. The generalization performance of the proposed HI-DKIELM algorithm is tested with ten sets of real UHI data sets, and compared with traditional ELM, I-ELM, EM-ELM, EI-ELM, and B-ELM.
6. In section “[Real-world application learning tasks](#)”, the performance of the HI-DKIELM in practical application learning tasks is verified and compared with other baseline methods.

Data and parameter settings

In these experiments, several databases are used in various experiments to verify the performance of the novel HI-DKIELM method.

For the experiment shown in sections “[Evaluation of the performance of the HCOBSO optimization algorithm using the CEC2017 and CEC2019 database](#)” and “[Performance evaluation of the HCOBSO optimization algorithm on typical functions](#)”, the CEC2017 database and five typical optimization functions are used to prove the optimization effectiveness of the proposed HCOBSO optimization method, respectively.

The CEC2017 database consists of a unimodal function ($F_1 - F_3$), multimodal function ($F_4 - F_{10}$), mixed-function ($F_{11} - F_{20}$), and hybrid function ($F_{21} - F_{30}$). The public parameters for all the algorithms are set the same to reflect the fairness of the experiment: the $D = 30$; the evaluation times of the Maximum function were $D * 10000$.

The CEC2019 database is known “The 100-Digit challenge” which are intend to be used in annual optimization competition. They are used to evaluate the algorithm for large scale optimization problems. The first three functions, CEC01–CEC03, have various dimensions. On the other hand, the CEC04 to CEC10 functions set as ten-dimensional minimization problems in the range $[-100, 100]$, and they are shifted and rotated. All the CEC functions are scalable and all global optimum of these functions were united toward point 1.

Table 3 Typical optimization functions

Function name	Equations
Sphere	$F_{n1} = \sum_{i=1}^D x_i^2$
Quadric	$F_{n2} = \left(\sum_{i=1}^D x_i^2 \right)$
Griewank	$F_{n3} = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}} + 1\right)$
Rosenbrock	$F_{n4} = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right)$
Rastrigin	$F_{n5} = \sum_{i=1}^D (x_i^2 - 10\cos(2\pi x_i) + 10)$

The five typical functions are shown in Table 3. The MaxDT are setting as 100 and 500 for $D = 10$ and $D = 30$, respectively. The dimension of the solution in every typical function D is 30, and the range of the solution of the typical function F_{n5} is setting as $[-100, 100]$, while the remains are setting as $[-30, 30]$.

In section “[Evaluation of the performance of the HCOBSO optimization algorithm using the CEC2017 and CEC2019 database](#)”, the best parameter is $N = 100$, $N_c = 5$, and $N_p = 20$ for the COA optimization method. The parameter of the HCOBSO is setting as $N = 100$, $N_c = 5$, $N_p = 20$ in the earlier searching, and the parameter is setting as $N = 100$, $N_c = 10$, and $N_p = 10$ in the later searching.

In sections “[Selection of hyperparameters](#)” and “[Regression analysis](#)”, the UHI machine data set is adopted in the experiment with 13 regression problems and 14 classification problems. The data specifications are shown in Table 4. The relevant data of the training set and test set are represented by the notation #train and the notation #test, respectively.

Evaluation of the performance of the HCOBSO optimization algorithm using the CEC2017 and CEC2019 database

To test the contribution of each improvement to the HCOBSO method, the HCOBSO method is compared with some incomplete optimization algorithms such as HCOBSO₅ ($N_c = 5$, $N_p = 20$), COA, BSO, ICOA, LPB [27], and FDO [28] on the CEC2017 and CEC2019 database, as shown in Table 5. In the experiment, three functions are selected in each group. The rank method is used in this experiment to compare the mean. The rank becomes better with the decrease of the mean. When the mean is the same, the standard variance will be compared.

Table 4 The specification of 27 benchmark UHI machine data sets

Datasets	Type	#Train	#Test
Auto MPG	Regression	200	192
Machine CPU	Regression	200	109
Fried	Regression	20,768	20,000
Wine Quality	Regression	2898	2000
Puma	Regression	4500	3692
California Housing	Regression	16,000	4000
House 8L	Regression	16,000	6784
Parkinsons motor	Regression	4000	1875
Parkinsons total	Regression	4000	1875
Puma	Regression	6000	2192
Delta elevators	Regression	6000	3000
Abalone	Regression	3000	1477
A9a	Regression	32,561	16,281
Colon-cancer	Classification	40	22
USPS	Classification	7291	2007
Sonar	Classification	150	58
Hill Valley	Classification	606	606
Protein	Classification	17,766	6621
Covtype binary	Classification	300,000	280,000
Mushrooms	Classification	4000	4122
Gisette	Classification	6000	1000
Leukemia	Classification	38	35
Duke	Classification	29	15
Connect-4	Classification	50,000	17,557
Mnist	Classification	40,000	30,000
DNA	Classification	1046	1186
W3a	Classification	4912	44,837

Table 5 shows that the HCOBSO₅ will get the best mean and variance for the most times on unimodal functions, indicating that the global searching ability of the BSO method and the mining ability of the COA algorithm are improved with the growth of N_p . The large value of N_p means the strong mining ability and fast rate of convergence. However, the HCOBSO₅ will fall into local minimum easily and the optimization performance is not well on multimodal functions. Among the five optimization methods, the average rank of the HCOBSO is 1.58, while the average rank of COA, BSO, ICOA, and HCOBSO₅ are 4.16, 4.58, 2.75, and 2, respectively. The HCOBSO optimization method proposed in this paper gets the best total rank among the five methods, while the total ranks of COA, BSO, ICOA, and HCOBSO₅ are 4, 5, 3, and 2, respectively. The results have shown that the HCOBSO can obtain the best optimization performance and prove the effectiveness of this method.

From the results shown in Table 6, we can find that the HCOBSO will get the best mean and variance for the most times on CEC 2019 database, that means with the growth

Table 5 The comparison results of HCOBSO and its incomplete algorithms under CEC2017 database

Function		Criterion	COA	BSO	ICOA	HCOBSO ₅	HCOBSO
Unimodal function	F_1	Mean	1.2099×10^3	1.2813×10^9	1.0737×10^2	4.1072×10^{-4}	7.4494×10^{-4}
		Std	1.2998×10^3	9.6388×10^8	1.0569×10^2	8.5916×10^{-4}	1.4801×10^{-4}
		Rank	5	4	3	1	2
	F_2	Mean	2.9013×10^{21}	3.1831×10^{32}	8.6764×10^{15}	4.8580×10^3	1.1941×10^1
		Std	1.1461×10^{22}	1.5894×10^{33}	3.5675×10^{16}	3.3985×10^4	2.4077×10^1
		Rank	4	5	3	2	1
	F_3	Mean	6.0573×10^4	2.8342×10^4	3.3032×10^4	7.6972×10^{-1}	9.5410×10^{-1}
		Std	1.0178×10^4	9.2323×10^3	6.8408×10^3	9.8032×10^{-1}	1.9288×10^0
		Rank	5	3	4	1	2
Multimodal function	F_4	Mean	8.4041×10^4	2.0825×10^2	4.8248×10^1	2.0841×10^1	1.8113×10^1
		Std	8.5306×10^0	8.4445×10^1	3.3517×10^1	3.0540×10^1	2.7696×10^1
		Rank	5	4	3	2	1
	F_5	Mean	5.2890×10^1	9.6116×10^1	3.4844×10^1	3.5884×10^1	2.8433×10^1
		Std	1.5025×10^1	3.2690×10^1	1.0983×10^1	1.0115×10^1	6.8876×10^2
		Rank	4	5	2	3	1
	F_6	Mean	1.6399×10^{-5}	6.3664×10^0	2.8782×10^{-4}	9.4452×10^{-7}	1.7483×10^{-7}
		Std	9.6428×10^{-6}	3.1596×10^0	1.6254×10^{-4}	2.4080×10^{-6}	4.7524×10^{-7}
		Rank	3	5	4	2	1
Mixed function	F_{11}	Mean	4.1143×10^1	4.9771×10^2	2.2685×10^1	2.9822×10^1	2.1678×10^1
		Std	2.6773×10^1	6.4235×10^2	2.0893×10^1	2.6125×10^1	2.0907×10^1
		Rank	4	5	2	3	1
	F_{12}	Mean	1.2532×10^5	4.0185×10^7	1.3660×10^5	1.2577×10^4	9.8943×10^3
		Std	1.2555×10^5	7.3849×10^7	9.3092×10^4	6.4424×10^3	6.0932×10^3
		Rank	3	5	4	2	1
	F_{13}	Mean	2.0357×10^4	2.8073×10^6	3.6293×10^2	3.0625×10^3	1.9749×10^3
		Std	2.6333×10^4	1.6225×10^7	8.9975×10^1	6.2901×10^3	3.8562×10^3
		Rank	4	5	1	3	2
Hybrid function	F_{21}	Mean	2.5626×10^2	2.8298×10^2	2.3724×10^2	2.2713×10^2	2.3023×10^2
		Std	1.6800×10^1	2.5684×10^1	1.1261×10^1	1.0815×10^1	8.5095×10^0
		Rank	4	5	3	1	2
	F_{22}	Mean	1.9999×10^3	8.0434×10^2	1.0005×10^2	1.0005×10^2	1.0010×10^2
		Std	1.5970×10^3	1.1113×10^3	3.4434×10^{-1}	2.4354×10^{-1}	4.8096×10^{-1}
		Rank	5	4	2	1	3
	F_{23}	Mean	4.1635×10^2	4.7029×10^2	3.8441×10^2	3.8706×10^2	3.7755×10^2
		Std	1.6898×10^1	2.9324×10^1	1.0543×10^1	1.3271×10^1	1.0911×10^1
		Rank	4	5	2	3	1

Best value in each row is indicated in bold

of N_p , the global searching ability of the BSO method and the mining ability of the COA algorithm are improved in the meantime correspondingly. That means the larger the value of N_p , the more strong mining ability and faster rate of convergence. Among the six optimization methods, the average rank of the HCOBSO is 1.7, while the average ranks of LPB, FDO, COA, BSO, and HCOBSO₅ are 3.3, 3, 4.5, 5.2 and 3.3 respectively. In the meantime, we can observe that the HCOBSO optimization method proposed in this paper gets the best total rank among the six methods.

Performance evaluation of the HCOBSO optimization algorithm on typical functions

In this part, we compare the HCOBSO algorithm with four baseline optimization methods: COA, BSO, ICOA, and DE-MPGWO [15] using typical functions.

In this study, the number of iterations of each algorithm is set to 2000, and the overall scale of the four baseline optimization methods is the same, i.e., $N_p = 40$. The experiments

Table 6 The comparison results of HCOBSO and its incomplete algorithms under CEC2019 database

Function	Criterion	LPB	FDO	COA	BSO	HCOBSO ₅	HCOBSO
CEC01	Mean	7.4943×10^9	4.5852×10^3	1.0284×10^4	1.0819×10^{10}	3.4911×10^{-5}	6.3319×10^{-4}
	Std	8.1382×10^9	2.0707×10^4	1.0138×10^4	7.5182×10^9	6.7014×10^{-5}	1.1544×10^{-4}
	Rank	5	3	4	6	1	2
CEC02	Mean	1.7638×10^1	4.0×10^0	2.4661×10^{20}	2.7384×10^{31}	5.9753×10^2	1.4687×10^0
	Std	3.1898×10^{-1}	3.224×10^{-9}	0.8939×10^{21}	1.3668×10^{32}	4.9618×10^3	3.5152×10^0
	Rank	3	2	5	6	4	1
CEC03	Mean	1.2702×10^3	1.3702×10^1	5.1487×10^5	2.4374×10^5	7.4675×10^{-2}	8.3960×10^{-2}
	Std	0	1.6490×10^{-11}	7.9338×10^4	8.1244×10^4	8.4479×10^{-2}	2.6748×10^{-1}
	Rank	4	3	6	5	1	2
CEC04	Mean	7.7908×10^1	3.4083×10^1	7.3956×10^3	1.7909×10^1	2.5634×10^0	2.2278×10^0
	Std	2.9885×10^1	1.6528×10^1	7.3363×10^{-1}	7.4311×10^0	4.4588×10^0	4.0436×10^0
	Rank	5	4	6	3	2	1
CEC05	Mean	1.1882×10^0	2.1392×10^0	4.6543×10^2	8.2659×10^2	4.4137×10^2	3.4972×10^{-1}
	Std	1.0945×10^{-1}	8.5751×10^{-2}	1.2821×10^2	2.8113×10^2	1.4767×10^2	9.5589×10^0
	Rank	2	3	5	6	4	1
CEC06	Mean	3.7389×10^0	1.2132×10^1	1.4431×10^{-6}	5.6024×10^{-1}	9.7568×10^{-6}	2.1504×10^{-8}
	Std	8.2305×10^{-1}	6.002×10^{-1}	8.2928×10^{-7}	2.7172×10^{-1}	3.5156×10^{-5}	6.9385×10^{-8}
	Rank	5	6	2	4	3	1
CEC07	Mean	1.4528×10^2	1.2048×10^2	3.6205×10^2	4.3798×10^3	3.6681×10^2	2.6639×10^1
	Std	1.7789×10^2	1.3593×10^1	2.3024×10^2	5.5242×10^3	3.8142×10^2	3.0524×10^1
	Rank	3	2	4	6	5	1
CEC08	Mean	4.8876×10^0	6.1021×10^1	1.1028×10^4	3.5362×10^6	1.5469×10^3	8.6984×10^{-1}
	Std	6.7942×10^{-1}	7.5699×10^{-1}	1.0793×10^4	6.3511×10^6	9.4059×10^2	8.8972×10^{-2}
	Rank	2	3	5	6	4	1
CEC09	Mean	2.8942×10^0	2.0×10^0	1.7914×10^5	2.4704×10^7	3.7686×10^1	2.4291×10^{-2}
	Std	2.3138×10^{-1}	1.5916×10^{-10}	2.2646×10^5	1.3953×10^8	9.1835×10^1	5.6353×10^{-2}
	Rank	3	2	5	6	4	1
CEC10	Mean	2.0001×10^1	2.7182×10^0	2.2551×10^1	2.4902×10^1	2.7936×10^1	2.8318×10^1
	Std	2.33×10^{-3}	8.8817×10^{-16}	1.4448×10^0	2.2088×10^0	1.5789×10^0	7.4883×10^{-1}
	Rank	1	2	3	4	5	6
Count		1	0	0	0	2	7
Ave rank		3.3	3	4.5	5.2	3.3	1.7
Total rank		3	2	5	6	3	1

Best value in each row is indicated in bold

are performed 50 times, and the average results are the final optimization result. Tables 7 and 8 show the optimization result of COA, BSO, ICOA, DE-MPGWO, and HCOBSO methods, while five typical functions are used under $D = 10$ and $D = 30$, respectively.

Tables 7 and 8 show the function F_{n2} have an ideal performance, and other results of the nine typical methods are summarized as follows: (1) the proposed HCOBSO presents more precise results compared with COA, BSO, ICOA, and DE-MPGWO methods when dealing with the other four typical functions. (2) The BSO algorithm can quickly drop to the minimum and jump out of the local minimum with a short time domain.

To sum up, the HCOBSO optimization method proposed in this study has a good balance in search of accuracy and convergence speed and can improve search ability.

Selection of hyperparameters

In this part, we will deduce parameters in HI-DKIELM and the original ELM scheme. Compared with the traditional neural network training algorithm, the ELM and HI-DKIELM need fewer parameters. According to the introduction of HI-DKIELM in section “The proposed HI-DKIELM”, two specified parameters are crucial: penalty factor C and the number of hidden nodes L . The results are shown in Fig. 4.

Table 7 Typical optimization function optimization results under $D = 10$

Function	Performance	COA	BSO	ICOA	DE-MPGWO	HCOBSO
F_{n1}	Mean	1.7833×10^2	9.0799×10^{-15}	3.4157×10^{-5}	0.9854×10^{-9}	6.0684×10^{-9}
	Variance	6.3524×10^1	2.4849×10^{-14}	2.4885×10^{-5}	5.8208×10^{-10}	4.8458×10^{-9}
F_{n2}	Mean	2.3737×10^0	1.3222×10^{-9}	1.3703×10^{-3}	2.0305×10^{-7}	8.4133×10^{-6}
	Variance	4.0946×10^{-1}	1.1382×10^{-9}	6.5582×10^{-4}	6.0356×10^{-7}	3.7531×10^{-6}
F_{n3}	Mean	1.6180×10^2	1.0000×10^{-1}	1.2593×10^{-2}	4.0517×10^{-4}	4.9377×10^{-7}
	Variance	5.0787×10^1	3.0513×10^{-1}	6.8779×10^{-2}	1.0714×10^{-3}	4.8665×10^{-7}
F_{n4}	Mean	4.0253×10^0	1.5325×10^{-6}	3.5760×10^{-7}	7.0882×10^{-4}	1.2498×10^{-10}
	Variance	1.6104×10^0	9.4192×10^{-7}	4.1539×10^{-7}	4.4363×10^{-6}	2.0501×10^{-10}
F_{n5}	Mean	7.1619×10^2	2.4093×10^{-5}	4.6770×10^{-6}	1.2678×10^0	2.0046×10^{-8}
	Variance	1.5819×10^3	1.4121×10^{-5}	5.6661×10^{-6}	7.8358×10^0	5.9686×10^{-8}

Best value in each row is indicated in bold

The learning accuracies of ELM and HI-DKIELM in the L subspace are shown in Fig. 4a and b; the parameter C is prefixed. The learning accuracies of ELM and HI-DKIELM in the C subspace are shown in Fig. 4c and d; the parameter L is prefixed. Figure 4a and b shows that the proposed HI-DKIELM follows a similar convergence property compared to the ELM but with higher testing accuracy. Meanwhile, the performance tends to be more stable in a wide range L . According to the results shown in Fig. 4c and d, the accuracy of HI-DKIELM changes slightly as C increases, and then increases rapidly to converge a better performance than the original ELM.

The value L needs to be large enough. Since L does not affect the accuracy of the curve, we can select C with few nodes.

Regression analysis

In this part, we will test the performance of the HI-DKIELM method proposed in section “The proposed HI-DKIELM” based on the regression analysis using the actual UHI data set in Table 4 to evaluate its generalization and robustness, compared with four baseline ELM methods: enhanced random search-based ELM (EI-ELM), error minimization ELM (EM-ELM), incremental ELM (I-ELM), and bidirectional ELM (B-ELM). The initially hidden layer neurons and increasing hidden neurons in the network are set as one in the experiment. The number of neurons in the hidden layer is the same as the number of iterations. In Table 9, we compare the testing root-mean-square error (RMSE) and training time of the regression problem.

The proposed method shows a similar generalization performance as other ELMs with much higher learning speed. For instance, the training time of the proposed HI-DKIELM method is only 0.0001S, 156 times as fast as that of ELM, 2025 times of I-ELM, 13004 times of EI-ELM, 75 times of EM-ELM, and 3732 times of B-ELM in the Auto MPG dataset. The training time of the proposed HI-DKIELM method is about 0.0001S, 616 times as fast as that of ELM, 4680 times of I-ELM, 35478 times of EI-ELM, 431 times of EM-ELM, and 453 times of B-ELM in the Delta elevators dataset. Furthermore, in this section, the performance of ELM and the proposed method has been tested in the XOR problem, which has one training sample in each class. The aim of this simulation is to verify whether the proposed method can handle some rare cases, such as cases with extremely few training data sets. The results are that the testing RSME obtained by ELM is close to 0.0010 when 2000 hidden nodes are used, while the testing RMSE obtained by the proposed method is close to 0.0010 when 87 hidden nodes are used.

In addition, the comparison results of average test accuracy are shown in Fig. 5. In Fig. 5, the x represents the number of hidden nodes, and the y represents the average testing root-mean-square error (RMSE). Compared with other

Table 8 Typical optimization function optimization results under $D = 30$

Function	Performance	COA	BSO	ICOA	DE-MPGWO	HCOBSO
F_{n1}	Mean	3.2554×10^1	5.4432×10^{-41}	7.2595×10^{-9}	1.1108×10^2	1.3966×10^{-17}
	Variance	5.9767×10^0	7.1605×10^{-41}	2.4885×10^{-5}	0.5141×10^1	3.2255×10^{-17}
F_{n2}	Mean	1.2998×10^0	6.0158×10^{-24}	5.3463×10^{-5}	1.4922×10^2	2.8862×10^{-10}
	Variance	1.9835×10^{-1}	6.2402×10^{-24}	3.9178×10^{-5}	0.8839×10^1	4.6435×10^{-10}
F_{n3}	Mean	3.3700×10^1	3.3333×10^{-2}	3.5149×10^{-2}	6.1414×10^{-4}	1.2484×10^{-7}
	Variance	7.9877×10^0	1.8257×10^{-1}	1.5826×10^{-2}	0.4517×10^{-4}	1.7135×10^{-7}
F_{n4}	Mean	3.8002×10^0	1.5129×10^{-2}	1.7828×10^{-2}	2.2663×10^3	1.0451×10^{-17}
	Variance	1.3167×10^0	1.0471×10^{-2}	3.9296×10^{-2}	1.6178×10^2	3.0738×10^{-17}
F_{n5}	Mean	1.7376×10^1	1.6787×10^{-1}	3.2962×10^{-3}	0.8385×10^{-4}	5.3309×10^{-17}
	Variance	5.8917×10^0	1.1943×10^{-1}	5.2111×10^{-3}	2.2097×10^{-5}	1.5184×10^{-16}

Best value in each row is indicated in bold

ELM methods, the HI-DKIELM method has the best test performance. For instance, the test error of the Ca house dataset obtained by the HI-DKIELM method is half of that of ELM. In addition, with 2000 hidden nodes, the test RMSE of the ELM method is close to 0.0010, while that of the HI-DKIELM method is close to 0.0010 with 89 hidden nodes.

Classification analysis

In this part, for the classification problem, we will test the performance of the HI-DKIELM method proposed in section “The proposed HI-DKIELM” using the actual UHI data set given in Table 4 to evaluate its generalization and robustness and compare it with the four baseline ELM methods.

The initially hidden layer neurons and a growing number of hidden neurons in the network are set as one in the experiment. The number of neurons in the hidden layer is the same as the number of iterations. We compare the testing root-mean-square error (RMSE) and training time of the regression problem. The comparison of the testing accuracy is displayed in Fig. 6.

The results shown in Fig. 6 shows that the proposed HI-DKIELM has significant advantages in training speed. In the Connect-4 dataset with a large number of medium input dimension training samples, the running speed of the proposed method is 5600 times, 300 times, 45 times, and 1000 times as fast as that of I-ELM, EI-ELM, EM-ELM, and PC-ELM, respectively. The moderate number of training samples in the Hill Valley dataset has a moderate input dimension. In the hill Valley dataset, the running time of the proposed method is 3 times, 20 times, 25 times, and 1200 times as fast as that of I-ELM, EI-ELM, EM-ELM, and PC-ELM, respectively, with better performance.

Real-world application learning tasks

In this section, we will evaluate the performance of the proposed HI-DKIELM on the real-world applications: image classification task.

The original data are color images from the Corel data set. Each image is segmented, using the Blob world system, into fragments that represent instances. Fragments containing specific visual content (e.g., elephant) are labeled positive, while the remaining fragments are labeled negative. Therefore, fragments (i.e., instances) from the same kind of image (e.g., elephant) form a binary learning problem. We have five different image data sets: Tiger, Elephant, Fox, Bikes, and Cars, and the number of instances is 1096, 1259, 1474, 5215, and 5600, respectively. The instances in Tiger, Elephant, and Fox data sets are described by a 230-dimensional feature vector which represents the color, texture, and shape of the region, while the instances in Bikes and Cars data sets are represented by a 90-dimensional feature vector.

Fig. 4 Testing accuracy in (C, L) subspace for the HI-DKIELM and ELM. **a** Accuracy of HI-DKIELM in terms of L . **b** Accuracy of ELM in terms of L . **c** Accuracy of HI-DKIELM in terms of C . **d** Accuracy of ELM in terms of C

Table 9 Generalization performance comparison between different ELM-based methods on regression problem

Datasets	ELM		I-ELM		EI-ELM		EM-ELM		B-ELM		Proposed	
	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time
Auto MPG	0.0976	0.0156	0.1000	0.2025	0.0918	1.3004	0.0968	0.0075	0.0920	0.3732	0.0896	0.0001
Machine CPU	0.0513	0.0069	0.0591	0.1909	0.0551	1.2633	0.0521	0.1385	0.0554	0.3469	0.0896	0.0001
Fried	0.0619	1.3135	0.1135	0.8327	0.0856	7.4016	0.0618	18.029	0.0857	5.5063	0.0734	0.0046
Wine Quality	0.1229	0.1950	0.1360	0.3516	0.1266	2.7126	0.1216	0.3806	0.1246	1.7098	0.1150	0.0027
Puma	0.1602	0.3725	0.1860	0.5070	0.1830	4.2161	0.1478	4.8392	0.1832	2.1846	0.1608	0.0102
House 8L	0.0718	0.8369	0.0946	1.1872	0.0850	10.7691	0.0763	7.0388	0.0818	3.8821	0.0719	0.0017
PD motor	0.2190	0.2730	0.2318	0.4639	0.2251	3.9016	0.2196	0.7394	0.2241	4.7680	0.1910	0.0035
PD total	0.2076	0.2838	0.2178	0.4678	0.2124	3.7854	0.2094	0.5944	0.2137	4.9278	0.2034	0.0021
Abalone	0.0824	0.0761	0.0938	0.3398	0.0848	2.6676	0.0817	0.1638	0.0808	1.2549	0.0708	0.0013
Delta elevators	0.0413	0.0616	0.0538	0.4680	0.0417	3.5478	0.0421	0.1342	0.0431	1.3946	0.0438	0.0001
Ca House	0.1354	0.9753	0.1801	1.1482	0.1505	12.0832	0.1302	3.5574	0.1450	7.2625	0.1263	0.0075

Best value in each row is indicated in bold

Fig. 5 The average testing accuracy of the different ELM based methods under different database for the regression problem

In image classification, visual content-based image retrieval is an important application, for example, finding pictures containing an elephant from a data set. In this

subsection, sample images from the benchmark data sets are shown in Fig. 7.

Fig. 6 The average testing accuracy of the different ELM based methods under different database for classification problem

The detailed experimental results are in Tables 10 and 11 for image classification tasks in terms of classification accuracy (ACC) and area under the curve (AUC), respectively. In Tables 10 and 11, the proposed HI-DKILEM on all image datasets achieves the best ACC and AUC, indicating HI-DKILEM is superior to other methods in content-based image retrieval tasks. The extraordinary performance is due to many local approximations created by the proposed HI-DKILEM. The results show that the naive Bayes (NB) method on the Tiger, Elephant, Bikes, and Cars datasets has the worst ACC and AUC performance. However, the SVM method on the Fox data set has the most inferior performance. For other baselines, more detailed experiment results can be found in Tables 10 and 11.

Conclusion

In this paper, a novel ELM method named Hybrid Intelligent Deep Kernel Incremental Extreme Learning Machine (HI-DKILEM) based on the hybrid coyote optimization Beetle Swarm Optimization (HCOBSO) method is proposed. The proposed method has several features different from existing ELM-based methods.

1. An improved Coyote Optimization Algorithm has been proposed to improve the searching efficiency and convergence, and the Gaussian global best-growing operator is utilized to replace the original growing operator.
2. An improved Beetle Swarm Optimization Method has been designed according to Tent mapping inverse learning and dynamic mutation strategies to prevent the BSO algorithm from falling into a local optimum.
3. A novel hybrid intelligent optimization algorithm called hybrid coyote optimization Beetle Swarm Optimization (HCOBSO) method for optimizing the parameters of the DKILEM has been presented. The proposed hybrid intelligent optimization method combines the global search ability of the coyote optimization algorithm and the local search capability of the Beetle Swarm Optimization algorithm.

In future work, there is developing space to explore the proposed methods. First, it is still challenging to make more insight into ELM for exploring its deep learning capability. Second, from an optimization point of view, we can use other novel heuristic algorithms, such as the five-element cycle optimization method.

Fig. 7 Example images used in the experiment from the COREL image categorization database

Table 10 Experimental results on image classification data sets concerning classification accuracy (ACC) %

Data set	Proposed	ELM	EN-ELM	W-ELM	SVM	KNN	NB
Tiger	80.51	76.78	78.05	76.91	67.67	77.31	60.74
Elephant	82.29	79.03	80.42	78.87	73.85	78.56	65.55
Fox	67.85	64.46	65.18	64.11	52.19	64.83	57.15
Bikes	77.00	76.29	76.49	75.65	67.78	71.77	61.63
Cars	63.53	62.36	62.14	61.77	57.36	61.47	55.17

Best value in each row is indicated in bold

Table 11 Experimental results on image classification data sets concerning the area under the ROC curve (AUC) (%)

Data set	Proposed	ELM	EN-ELM	W-ELM	SVM	KNN	NB
Tiger	88.43	84.00	86.59	84.46	65.07	77.15	61.05
Elephant	90.33	87.12	89.08	87.05	71.83	78.44	66.23
Fox	74.87	70.05	71.51	69.74	53.56	64.87	57.44
Bikes	83.65	82.59	82.43	82.35	67.85	71.73	61.64
Cars	69.20	67.41	67.26	66.83	56.23	61.26	55.93

Best value in each row is indicated in bold

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive comments.

Funding This work were supported by National Natural Science Foundation of China (Grant no. 62006075), National Natural Science Foundation of Hunan Province of China (Grant no. 2019JJ50106), Hunan Provincial Science and Technology Department (CN) (Grant no. 19A117) and the science and technology innovation program of Hunan Province (Grant no. 2020RC5019).

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. *Neurocomputing* 70(1):489–501
- Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feed forward networks with random hidden nodes. *IEEE Trans Neural Netw* 17(4):879–892
- Zeng Y, Li Y, Chen J et al (2020) ELM embedded discriminative dictionary learning for image classification. *Neural Netw* 123:331–342
- Dasgupta S, Stevens CF (2017) A neural algorithm for a fundamental computing problem. *Science* 358(6364):793–796
- Wang W, Zhang R (2014) Improved convex incremental extreme learning machine based on enhanced random search. *Electr Eng Electron Eng* 238:2033–2040
- Tang J, Deng C, Huang G-B (2016) Extreme learning machine for multilayer perceptron. *IEEE Trans Neural Netw Learn Syst* 27(4):809–821
- Zhang R, Lan Y, Huang GB et al (2012) Universal approximation of extreme learning machine with adaptive growth of hidden nodes. *IEEE Trans Neural Netw Learn Syst* 23:365–371
- Cui D, Huang G, Kasun LLC, Zhang G, Han W (2017) Elmnt: feature learning using extreme learning machines. In: 2017 IEEE international conference on image processing (ICIP) (pp 1857–1861)
- Foroughi H, Ray N, Zhang H (2018) Object classification with joint projection and low-rank dictionary learning. *IEEE Trans Image Process* 27(2):806–821
- Wang S, Zhu E, Yin J, Porikli F (2018) Video anomaly detection and localization by local motion based joint video representation and OCELM. *Neurocomputing* 277:161–175
- Sun K, Mou S, Qiu J, Wang T, Gao H (2018) Adaptive fuzzy control for non-triangular structural stochastic switched nonlinear systems with full state constraints. *IEEE Trans Fuzzy Syst* 2018:1
- Zhang W, Zhanga Z, Wang L et al (2019) Extreme learning machines with expectation kernels. *Pattern Recogn* 96:1–13
- Lekamalage CL et al (2016) Dimension reduction with extreme learning machine. *IEEE Trans Image Process* 25(8):3906–3918
- Yang Y, Wu QMJ (2016) Extreme leaning machine with subnetwork hidden nodes for regression and classification. *IEEE Trans Cybern* 46(12):2885–2898
- Wu D, Qu ZS, Guo FJ, Zhu XL, Wan Q (2019) Hybrid intelligent deep kernel incremental extreme learning machine based on differential evolution and multiple population gray wolf optimization methods. *Automatika* 60(1):48–57
- Wu D, Qu ZS, Guo FJ, Wan Q (2019) Hybrid multilayer incremental hybrid cost sensitive extreme learning machine with multiple hidden output matrix and subnetwork hidden nodes. *IEEE Access* 7:118422–118434
- Scardapane S, Comminiello D, Scarpiniti M, Uncini A (2015) Online sequential extreme learning machine with kernels. *IEEE Trans Neural Netw Learn Syst* 26(9):2214–2220
- Li S, Song S, Huang G, Wu C (2019) Cross-domain extreme learning machines for domain adaptation. *IEEE Trans Syst Cybern Syst* 49(6):1–14
- Yu K, Liang J, Qu B, Luo Y, Yue C (2020) Dynamic selection preference-assisted constrained multiobjective differential evolution. *IEEE Trans Syst Man Cybern Syst*. <https://doi.org/10.1109/TSMC.2021.3061698>
- Lakhdhar W, Mzid R, Khalgui M, Li Z, Frey G, Al-Ahmari A (2019) Multiobjective optimization approach for a portable development of reconfigurable real-time systems: from specification to implementation. *IEEE Trans Syst Man Cybern Syst* 49(3):623–637
- Chen Q, Ding J, Yang S, Chai T (2020) A novel evolutionary algorithm for dynamic constrained multiobjective optimization problems. *IEEE Trans Evol Comput* 24(4):792–806
- Zhou Y, Xiang Y, He X (2021) Constrained multi-objective optimization: Test problem construction and performance evaluations. *IEEE Trans Evol Comput* 25(1):172–186. <https://doi.org/10.1109/TEVC.2020.3011829>
- Yu K, Liang J, Qu B, Yue C (2021) Purpose-directed two-phase multiobjective differential evolution for constrained multiobjective optimization. *Swarm Evol Comput* 60:100799
- Yu K, Chen X, Wang X, Wang Z (2017) Parameters identification of photovoltaic models using self-adaptive teaching-learning-based optimization. *Energy Convers Manag* 145:233–246
- Pierezan J, Coelho LDS (2018) Coyote optimization algorithm: a new metaheuristic for global optimization problems. In: 2018 IEEE congress on evolutionary computation (CEC). Rio de Janeiro, Brazil, IEEE
- Ling T, Zhan Z-H, Wang YX, Wang ZJ, Yu WJ, Zhang J (2018) Competitive swarm optimizer with dynamic grouping for large scale optimization. In: Proc. IEEE Congr. Evol. Comput. (CEC 2018), Rio de Janeiro, Brazil, pp 2655–2660
- Rahman CM, Rashid TA (2020) A new evolutionary algorithm: learner performance based behavior algorithm. *Egypt Inf J*. <https://doi.org/10.1016/j.eij.2020.08.003>
- Abdullah JM, Ahmed T (2019) Fitness dependent optimizer: inspired by the bee swarming reproductive process. *IEEE Access* 7:43473–43486. <https://doi.org/10.1109/ACCESS.2019.2907012>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.