Prog Artif Intell
DOI 10.1007/s13748-016-0102-4

REGULAR PAPER

A novel artificial bee colony optimization strategy-based extreme learning machine algorithm

Yang Wang[1,2] · Anna Wang[1] · Qing Ai[1] · Haijing Sun[1]

Received: 5 June 2016 / Accepted: 19 September 2016

Abstract Extreme learning machine (ELM) is a kind of single-hidden layer feedforward neural networks (SLFNs). Compared with traditional neural networks algorithms, ELM is simpler in structure with higher learning speed and better generalization performance. Due to generating randomly input weights and biases of ELM, there can exist some non-optimal or unnecessary input weights and biases. In addition, ELM can need more hidden nodes which can make ELM respond slowly to unknown testing data. Consequently, a new NABC-ELM algorithm, which is optimized by a novel artificial bee colony called NABC, is proposed. To improve generalization performance of ELM, the NABC is applied to optimize input weights and biases. In NABC, the Tent chaotic opposition-based learning method is applied to initialize the population. Meanwhile, the self-adaptive search strategy is presented in the employed bee and onlooker bee phase. In addition, the Tent chaotic local search for scout bee is implemented. Finally, experiments on some popular classification data sets demonstrate that the proposed NABC-ELM can consistently get better generalization performance than some existing ELM variants.

Keywords Extreme learning machine · Neural networks · Artificial bee colony · Chaotic opposition-based learning · Self-adaptive search · Chaotic local search

✉ Yang Wang
wangyang0531@163.com

1 College of Information Science and Engineering, Northeastern University, Shenyang 110004, People's Republic of China

2 School of Computer and Communication Engineering, Liaoning Shihua University, Fushun 113001, People's Republic of China

1 Introduction

Artificial neural networks (ANNs) have the ability of good generalization and non-linear mapping [1]. In the recent decades, ANNs have been widely applied in various fields, including pattern recognition, machine learning, image processing and automatic control, and so on [2]. As one of the most popular neural networks models, back propagation (BP) usually adopts gradient-based error back propagation algorithms [3]. In addition, support vector machine (SVM) is based on statistical learning and structural risk minimization principle [4]. However, these techniques need to set many parameters, take a long training time, and can easily get trapped into local optima. To address these issues, in 2004, Huang et al. proposed the extreme learning machine (ELM) technique which is a kind of single-hidden layer feedforward neural networks (SLFNs) [5]. In ELM, the weights from input layer to hidden layer and biases of hidden nodes are randomly generated instead of being iteratively learned. Moreover, the weights from hidden layer to output layer are analytically calculated by the least square method. Compared with BP and SVM, ELM provides better generalization performance at much faster learning speed and with least human intervenes [6]. The ELM approach has showed its superiority in various fields of applications [7,8]. However, due to generating randomly input weights and biases of ELM, there can exist some non-optimal or unnecessary input weights and biases. In addition, ELM can need more hidden nodes which can make ELM respond slowly to unknown testing data. For these problems, it is crucial that various optimization methods should be employed to adjust input weights and biases of ELM, and optimize the network structure.

Until now, biological-inspired optimization algorithms have been developed to be successful in tackling different kinds of optimization problems, such as particle swarm opti-

Published online: 04 October 2016

mization (PSO) [9], ant colony optimization (ACO) [10], genetic algorithm (GA) [11], artificial bee colony algorithm (ABC) [12], and so on. ABC algorithm based on simulating the foraging behavior of honey bee swarm was developed by Karaboga in 2005. Some researches have demonstrated that performance of ABC is superior to other methods [13,14]. In view of its simplicity and ease of implementation, ABC has attracted much attention and been widely applied in many real-world optimization problems [15–19]. However, ABC is good at exploration but poor at exploitation which results in poor convergence. Consequently, many ABC variants have been proposed to improve performance of ABC. For instance, Karaboga and Basturk proposed a modified ABC by controlling the frequency of perturbation and introducing the ratio of the variance operator [20]. Motivated by PSO, Zhu proposed a gbest-guided ABC (GABC) which makes use of the information of global best solution to improve the exploitation [21]. Gao and Liu proposed a modified ABC (MABC) using a modified search equation together with a novel chaotic initialization [22]. Li et al. introduced an inertia weight and two acceleration coefficients [23]. Yurtkuran and Emel proposed an adaptive ABC (AABC) using various search strategies within an overall search process [24]. Maeda and Tsuda presented a reduction of ABC algorithm for avoiding local minima [25]. Gao et al. developed a novel ABC with multiple search strategies (MuABC), and MuABC used an adaptive selection mechanism to produce candidate solutions [26]. Without question, these studies are very helpful to improve performance of ABC. Hence, searching for a well-improved optimization method is very necessary.

In this paper, a new NABC-ELM algorithm, which is optimized by a novel artificial bee colony called NABC, is proposed. To improve generalization performance of ELM, the NABC is applied to optimize input weights and biases. In NABC, the initial population is generated combining the Tent chaotic map with the opposition-based learning method [27,28], which may increase the quality of solution. Meanwhile, the self-adaptive search strategy is presented in the employed bee and onlooker bee phase to enhance convergence ability. In addition, the Tent chaotic local search for scout bee is implemented to jump out of local optima. Finally, experiments on some popular classification data sets demonstrate that the proposed NABC-ELM can consistently get better generalization performance than some existing ELM variants.

The rest of this paper is organized as follows. Section 2 briefly describes the backgrounds related to ELM and ABC. Section 3 presents the research works on proposed NABC algorithm. Section 4 describes NABC-ELM model in detail. Section 5 analyzes the performance of NABC-ELM through experiments on some popular classification data sets. Section 6 provides a conclusion for this paper.

2 Backgrounds

In this section, we briefly review the related works regarding ELM and ABC.

2.1 ELM

Given a training data set consisting of N arbitrary samples (s_j, t_j), where $s_j = [s_{j1}, s_{j2}, \ldots, s_{jn}]^T \in R^n$, and $t_j = [t_{j1}, t_{j2,} \cdots t_{jm}]^T \in R^m$. The jth sample s_j is an n×1 feature vector, and t_j is an $m \times 1$ target vector. Given hidden nodes $L \ll N$ and activation function g (x), then the standard mathematical model of SLFNs is as follows:

$$\sum_{i=1}^{L} \beta_i g\left(w_i \cdot s_j + \mathrm{b_i}\right) = \mathrm{t_j}, \quad j = 1, 2, \ldots, N \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the input weight vector connecting input nodes and the ith hidden node, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the output weight vector connecting the ith hidden node and the output nodes, b_i is the bias of the ith hidden node, and $w_i \cdot s_j$ is the inner product of w_i and s_j.

If the number of hidden nodes L is equal to the number of training samples N, then SLFNs can approximate the training samples with zero error. Equation (1) can compactly be rewritten in matrix form as:

$$H\beta = T \tag{2}$$

$$H = \begin{bmatrix} g(w_1 \cdot s_1 + b_1) & \cdots & g(w_L \cdot s_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot s_N + b_1) & \cdots & g(w_L \cdot s_N + b_L) \end{bmatrix}_{\mathrm{N\times L}}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L\times m}, \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N\times m} \tag{3}$$

where β is the output weight matrix, and T is the output matrix. H is the hidden layer output matrix, where the jth column of H represents the jth hidden node output vector in regard to all the inputs.

However, in most cases, it is $L \ll N$ and there may not exist a β that satisfies Eq. (2). The input weights and hidden layer biases need not be adjusted at all and can be randomly generated, so the output weights can be determined by finding the least square solution $\hat{\beta}$ of the linear system $H\beta = T$:

$$\begin{aligned} &\|H(w_1, \ldots, w_L, b_1, \ldots, b_L)\hat{\beta} - T\| \\ &= \min\|H(w_1, \ldots, w_L, b_1, \ldots, b_L)\beta - T\| \\ &\hat{\beta} = H^{+}T \end{aligned} \tag{4}$$

where H^+ is the Moore–Penrose generalized inverse of matrix H. If the H is non-singular, Eq. (4) can be calculated as:

$$\hat{\beta} = \left(H^T H\right)^{-1} H^T T. \tag{5}$$

The learning algorithm of ELM is summarized in Algorithm 1.

Algorithm 1 Given a training data set (s_j, t_j), including N distinct samples, L hidden nodes, and activation function g(x):

Step 1. Randomly generate input weights w_i and biases $b_i, i = 1, 2, \ldots, L$.
Step 2. Calculate the hidden layer output matrix H by Eq. (3).
Step 3. Calculate the output weight β by Eq. (5).

2.2 ABC

In ABC, the colony of artificial bee consists of three groups of bees: employed bees, onlooker bees, and scout bees. Half of the colony consists of the employed bees, and the other half consists of the onlookers. Employed bees take charge of searching available food sources and gathering required information, then they share their food information with onlooker bees. The onlookers choose good food sources from those found by the employed bees to further exploit the food sources. When the quality of the food source cannot be improved over the predefined number of cycles, the food source is abandoned by its employed bee, and the corresponding employed bee becomes a scout bee. Then, the scout bee randomly searches for a new food source to replace the abandoned food source.

In the initialization step, ABC generates a randomly distributed initial population of SN solutions, where SN denotes the number of employed or onlooker bees. The position of each food source X_i corresponds to a possible solution, where $X_i = \left\{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\right\}$ is a D-dimensional vector, and D is the number of optimization parameters. Each initial solution X_i is generated randomly within the range of the boundaries of the parameters as follows:

$$x_{i,j} = x_{\min,j} + \text{rand}\,(0, 1)\left(x_{\max,j} - x_{\min,j}\right) \quad i = 1, 2, \ldots, SN, \quad j = 1, 2, \ldots, D, \tag{6}$$

where $x_{\max,j}$ and $x_{\min,j}$ are the upper and lower bounds for the dimension j, respectively.

After the initialization, the population repeats cycles with the search processes of the employed bees, onlooker bees, and scout bees. In the employed bee phase, each employed bee generates a candidate position V_i by performing a local search around each food source as follows:

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right) j \epsilon \{1, 2, \ldots, D\}, \quad k \epsilon \{1, 2, \ldots, \text{SN}\}, \tag{7}$$

where j and k are randomly selected indexes, and $k$is different from i. $\phi_{i,j}$ is a random real number in the range of [−1, 1]. Then, the greedy selection is applied between X_i and V_i to retain the better food source.

In the onlooker bee phase, an onlooker bee selects a food source depending on the probability value p_i associated with that food source, and p_i is calculated as follows for the minimum value problem:

$$p_i = 1 - \frac{f_i}{\sum_{j=1}^{SN} f_j}, \tag{8}$$

where f_i denotes the fitness value of solution X_i. As a matter of fact, food sources with better fitness values will be more likely to be selected and updated. Once the onlooker bee selects the food source, a new food source position V_i is generated by Eq. (7). At the same time, the greedy selection is carried out again.

In the scout bee phase, if a food source can not be improved over the predefined number of cycles named *limit*, the employed bee abandons the food source. Whereafter, the corresponding employed bee becomes a scout bee, and the scout bee will randomly initialize a new food source by Eq. (6).

According to the analysis mentioned above, ABC algorithm is summarized in Algorithm 2.

Algorithm 2. Set the population size *SN*, the number of maximum cycles *MCN*, and the control parameter *limit*:

Step 1. Initialization phase:

Step 1.1 Randomly generate *SN* individual in the search space to form initial population;

Step 1.2 Evaluate fitness values of population;

Step 1.3 Set *iter*=1, and *iter* denotes the current iteration number.

Step 2. While *iter*<=*MCN*, do

Step 2.1 The employed bee phase;

Step 2.2 The onlooker bee phase;

Step 2.3 The scout bee phase;

Step 2.4 Memorize the best solution achieved so far, and update *iter*=*iter*+1.

Step 3. Output the best solution achieved.

3 A novel artificial bee colony algorithm (NABC)

To enhance convergence ability of ABC algorithm, a novel artificial bee colony algorithm called NABC is proposed, and it is described in detail as follows.

3.1 Chaotic opposition-based learning initialization method

Population initialization is a crucial task in ABC, because it can affect the convergence speed and the quality of the final solution. With the characteristics of the ergodicity, randomicity, and regularity, chaos cannot repeatedly traverse all states in a certain range according to their own rules. Therefore, the chaotic map is suitable to initialize the population to increase the population diversity and improve convergence ability [29,30]. However, different chaotic mapping operators have a great influence on optimization process. At present, the logistic map is quoted greatly in the literature [31]. Meanwhile, the Tent map shows better ergodic uniformity and higher search speed than the logistic map [27]. Therefore, in this study, the Tent map is used in chaotic initialization to generate the chaotic sequence. Tent map is defined as follows:

$$cx_{k,j} = \begin{cases} 2cx_{k-1,j} & 0 \le cx_{k-1,j} \le 1/2 \\ 2\left(1 - cx_{k-1,j}\right) & 1/2 < cx_{k-1,j} \le 1 \end{cases}$$
$$k = 1, 2, \ldots, \text{CM}, \; j = 1, 2, \ldots, D, \tag{9}$$

where $cx_{k,j}$ and $cx_{k-1,j}$ represent the jth position component in the kth chaotic iteration and in the $(k-1)$th chaotic iteration, respectively. CM is the maximum number of chaotic iteration.

In addition, opposition-based learning (OBL) proposed by Tizhoosh in 2005 is a new concept in computational intelligence. In OBL, the key principle is that the opposite solution is calculated and evaluated for a candidate solution. Set $X_i = \{x_{i,1}, x_{i,2}, \ldots x_{i,j}, \ldots, x_{i,D}\}, x_{i,j} \epsilon \left[a_j, b_j\right]$. Then, corresponding opposite solution $OX_i = \{ox_{i,1}, ox_{i,2}, \ldots, ox_{i,j}, \ldots, ox_{i,D}\}$ is defined as follows:

$$ox_{i,j} = a_j + b_j - x_{i,j}. \tag{10}$$

OBL has been proved to be an effective method to accelerate convergence speed and get better food source [28]. Hence, this paper combines the Tent chaotic map with the OBL method to generate initial population. The pseudocode of chaotic opposition- based learning initialization algorithm is described in Algorithm 3.

Algorithm 3. Set the population size *SN*, the dimension of a food source *D*, the maximum number of chaotic iteration *CM*:

Step 1. Chaos phase:

```
1)  for i=1: SN
2)      for j=1: D
3)  cx_{0,j} = rand(0,1)  except 0.2,0.4,0.6, 0.8
% To avoid falling into the small cycle
4)      for k=1: CM
5)          if cx_{k-1,j} ≤ 1/2
6)                cx_{k,j} = 2cx_{k-1,j}
7)          else    cx_{k,j} = 2(1 − cx_{k-1,j})
8)          end
9)          if   cx_{k,j} = {0,0.25,0.5,0.75}
% It denotes cx_{k,j} falls into the fixed point
10)               cx_{k,j} = cx_{k,j} + ε
11)         end
12)      end
13)      x_{i,j} = a_j + cx_{k,j}(b_j − a_j)
14)     end
15)  end
```

Step 2. OBL phase:

```
1)  for i=1: SN
2)      for j=1: D
3)          ox_{i,j} = a_j + b_j − x_{i,j}
4)      end
5)  end
```

Step 3. Selecting *SN* individuals with best fitness values from $\{X(SN) \cup OX(SN)\}$ as initial population.

3.2 Self-adaptive search strategy

It is well known that the balance between exploration and exploitation is crucial to the success of ABC. The exploration refers to the ability to search for the various unknown regions in the solution space, while the exploitation refers to the ability to apply the knowledge of previous solutions to find better solutions. In ABC, a new candidate solution is generated with the guidance of solution randomly selected by Eq. (7). However, there is no guarantee that solution randomly selected is a good solution, so the new candidate solution cannot be better than previous solution. However, the solution found by Eq. (7) is random enough for exploration. Therefore, ABC is good at exploration but poor at exploitation which results in poor convergence [21].

To improve performance of ABC, related research on search equations has been suggested [21,23,24,26,32,33]. Among those, the GABC proposed by Zhu is the most representative one. In GABC, a modified search equation with the guidance of the best solution is presented as follow:

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right) + \psi_{i,j}\left(x_{\text{best},j} - x_{i,j}\right), \quad (11)$$

where the third term in the right-hand side of Eq. (11) is a new added term. $\psi_{i,j}$ is a random real number in the range of [0, 1.5], and $x_{\text{best},j}$ is the jth element of the global best solution. However, the improvement is not remarkable based on the experimental results reported [21].

As a result, the self-adaptive search strategy is proposed in this paper. The search strategy plays an important role in determining performance of ABC. If only one search strategy is adopted at each generation, search ability may be limited. Although Eq. (7) avoids plunging into the local optima to some extent, it greatly degrades convergence speed. In addition, it has been observed that good solutions are suitable to be modified with the guidance of solution randomly selected, so as to get out of the local optima, while bad solutions are suitable to be modified with the guidance of the best solution, so as to accelerate convergence speed. As a consequence, two different search equations based on different emphases are presented as follows:

$$v_{i,j} = x_{\text{best},j} + \phi_{i,j}\left(x_{\text{best},j} - x_{k,j}\right) \quad (12)$$

$$v_{i,j} = x_{r1,j} + \phi_{i,j}\left(x_{i,j} - x_{r2,j}\right) j\epsilon\{1,2,\ldots,D\}, \quad r1, r2\epsilon\{1,2,\ldots,\text{SN}\}, \quad (13)$$

where $x_{\text{best},j}$ is the jth element of the global best solution, $r1$ and $r2$ are distinct integer randomly selected and are also different from i, and j is randomly selected index. $\phi_{i,j}$ is a random real number in the range of $[-1, 1]$.

Equation (12) can drive the candidate solution toward the best solution to accelerate convergence speed. The first term in the right-hand side of Eq. (13) represents X_{r1}, which is a randomly selected solution from the population. The search equation can bring more information and generate a more promising candidate solution. In a word, compared Eqs. (12) with (13), the former emphasizes the exploitation, while the latter emphasizes the exploration. On the other hand, self-adaptive selection mechanism based on a probabilistic selection method is implemented as follows:

$$\text{pf}_i = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}}, \quad (14)$$

where f_i is the fitness value of X_i, $f_{\max}$ and $f_{\min}$ are the maximal and minimal fitness value of the population, respectively. That is, according to self-adaptive selection mechanism, during the search process, different search rules are adopted to maintain the balance between exploration and exploitation. Based on the above analysis, the self-adaptive search strategy is illustrated in Algorithm 4.

Algorithm 4. Given solution X_i:

Step 1. Calculate the probability value pf_i by Eq. (14);

Step 2. if rand (0, 1)< pf_i

generate the candidate solution by Eq. (12)

else generate the candidate solution by Eq. (13)

3.3 Chaotic local search for scout bee

In ABC, if the food source position corresponding a possible solution cannot be improved over a predefined number of cycles named *limit*, it means that the solution has get trapped into local optima. Then, the new solution will be randomly generated and can affect convergence speed due to the randomicity. Chaos is a good search mechanism easy to jump out of local optima. To enhance the local search ability, chaotic local search for scout bee is employed to search a number of neighborhood points around local optima and converge quickly to the global optima. Based on the Tent map, chaotic local search for scout bee is specifically described in Algorithm 5.

Algorithm 5. Given the maximum number of chaotic iteration *CM*, the solution of search stagnation $X_g = \{x_{g,1}, x_{g,2}, \cdots, x_{g,D}\}$:

Step 1. Set the iterative variable *k*=1, and randomly generate a chaotic variable CX_0 in the range of [0, 1], where $CX_0 = \{cx_{0,1}, cx_{0,2}, \cdots, cx_{0,D}\}$ except 0.2, 0.4, 0.6, 0.8 ;

Step 2. Based on Tent map,calculate chaotic variable $CX_k = \{cx_{k,1}, cx_{k,2}, \cdots, cx_{k,D}\}$ $(k = 1,2,\cdots,CM)$ by Eq.(9);

Step 3. If $cx_{k,j} = \{0, 0.25, 0.5, 0.75\}$ $(j = 1,2,\cdots D)$

$cx_{k,j} = cx_{k,j} + \varepsilon$

Step 4. Map chaotic variables $cx_{k,j}$ to decision variables $v_{k,j}$ as follows:

$$v_{k,j} = x_{g,j} + R_j\left(2cx_{k,j} - 1\right) \quad (15)$$

Where R_j is chaotic search radius, and $v_{k,j}\epsilon[x_{g,j} - R_j, x_{g,j} + R_j]$;

Step 5. Calculate $f(V_K)$ and retain the best solution;

Step 6. Set *k*=*k*+1, if the maximum number of chaotic iteration is reached, chaotic local search is completed. Otherwise, go to step 2.

4 The proposed NABC-ELM model

ELM is a good method to learn with higher learning speed and better generalization performance. However, randomly generating input weights and biases easily gives rise to overfitting problem. In some practical applications, the problem has caused lower predicting accuracy. Therefore, it is necessary to develop a more effective learning method. This paper presents a new NABC-ELM model which adopts a novel artificial bee colony called NABC to optimize input weights and biases of ELM. Then, NABC-ELM can achieve higher accuracy and better stability than other learning methods.

Generally, data samples are divided into training samples and testing samples. Training samples are trained to build the classification model. Then, testing samples are test by classification model achieved. However, to avoid overfitting problem, this paper adopts k-fold cross validation (CV) method. In k-fold CV, data samples are divided into approximately the same size and mutually disjoint subsets, such as $A_1, A_2, \ldots, A_k$, and then, training and testing are performed for k iteration. In the ith iteration, A_i is selected as testing samples, while the rest of subsets are selected as training sample. Finally, classification results adopt average value of results for k iteration.

4.1 Design of individual form

Each individual of the population consists of a set of input weights and hidden biases, and specific form is as follows:

$$\theta = \{w_{11}, w_{12}, \ldots, w_{1L}, w_{21}, w_{22}, \ldots w_{2L}, \ldots, w_{n1}, w_{n2}, \ldots, w_{nL}, b_1, b_2, \ldots, b_L\}, \tag{16}$$

where w_{ij} and b_j are input weights and hidden biases in the range of $[-1, 1]$, respectively. Then, n and $L$are the numbers of input and hidden nodes. The dimension of each individual θ is $(n+1) \times L$.

4.2 Design of fitness value

The fitness value of each individual is defined as misclassification rate of training samples, and specific form is as follows:

$$f(\theta) = \frac{N_{\text{misclassification}}}{N_{\text{tr}}}, \tag{17}$$

where N_{tr} is the number of training samples, and $N_{\text{misclassification}}$ is the number of misclassification. However, higher training accuracy does not necessarily guarantee better testing accuracy. It has been observed that the smaller the norm of output weights is, the better generalization performance is [34]. Therefore, when the fitness values of different solutions are approximately equal, the solution with the smaller norm of output weights is selected as the better solution. The determination of new population is described as follows:

$$X_{lbest} = \begin{cases} X_i & \text{if } f(X_{\text{lbest}}) - f(X_i) > \lambda f(X_{\text{lbest}}) \\ X_i & \text{if } |f(X_{\text{lbest}}) - f(X_i)| < \lambda f(X_{\text{lbest}}) \\ & \&\& \|\beta_{X_i}\| < \|\beta_{X_{\text{lbest}}}\| \\ X_{\text{lbest}} & \text{else} \end{cases}$$
$$X_{\text{gbest}} = \begin{cases} X_i & \text{if } f\left(X_{\text{gbest}}\right) - f(X_i) > \lambda f\left(X_{\text{gbest}}\right) \\ X_i & \text{if } \left|f\left(X_{\text{gbest}}\right) - f(X_i)\right| < \lambda f\left(X_{\text{gbest}}\right) \\ & \&\& \|\beta_{X_i}\| < \|\beta_{X_{\text{gbest}}}\| \\ X_{\text{gbest}} & \text{else} \end{cases}, \tag{18}$$

where $f(X_i)$, $f(X_{\text{lbest}})$, and $f\left(X_{\text{gbest}}\right)$ are the fitness value of solution X_i, the best fitness value of the current iteration, and the global best fitness value, respectively. β_{X_i} is the output weights of solution X_i, $\beta_{X_{\text{lbest}}}$ is the output weights of solution X_{lbest}, $\beta_{X_{\text{gbest}}}$ is the output weights of solution X_{gbest}, and λ is tolerance rate.

Based on the above considerations, the pseudocode of the ABC-ELM model is given below.

Algorithm 6. The ABC-ELM model.

% Performance estimation by using k-fold CV where k=10

1) begin
2) for j=1:k
3) Training set=k-1 subsets
4) Testing set=remaining subset
5) Achieve a set of input weights and hidden biases using **Algorithm 2;**
6) Train ELM on the training set using **Algorithm 1;**
7) Test the trained ABC-ELM model on the testing set;
8) end
9) Return the average classification accuracy of ABC-ELM over jth testing set;
10）end

To achieve better generalization performance, in this paper, the NABC is firstly applied to optimize input weights and biases. Then, ELM is trained with these parameters achieved. Finally, the optimized model called NABC-ELM is used to predict testing samples, and achieve the final classification results. The proposed NABC-ELM model is specifically described in Algorithm 7.

Algorithm 7. The NABC-ELM model.

Step 1. Initialization phase:

Step 1.1 Given a training dataset (s_j, t_j), including N distinct samples, L hidden nodes, and activation function g(x);

Step 1.2 Ten-fold CV is used to divide data samples into ten subsets, of which nine are used as training samples, and the remaining one is used as testing samples;

Step 2. Training phase:

Step 2.1 According to **Algorithm 3**, Generate *SN* individual in the search space to form initial population by Eq.(16), and regard each individual as a food source position;

Step 2.2 Evaluate fitness values of population by Eqs.(17) and (18), and set *iter*=1;

Step 2.3 Set *trial(i)*= 0, where $i = 1,2,\cdots,SN$, and *trail(i)* denotes the unimproved number of solution X_i.

Step 2.4 The employed bee phase:

Step 2.4.1 According to **Algorithm 4**, Generate a candidate solution V_i;

Step 2.4.2 Calculate $f(V_i)$:

if $f(V_i) < f(X_i)$

$X_i = V_i$; $trial(i)$=0

else $trial(i) = trial(i)$+1

Step 2.5 calculate the probability value p_i by Eq.(8);

Step 2.6 The onlooker bee phase: set *t*=0, *i*=1;

while *t*<=*SN*, do

if rand(0,1)< p_i

Step 2.6.1 According to **Algorithm 4**, Generate a candidate solution V_i;

Step 2.6.2 Calculate $f(V_i)$:

if $f(V_i) < f(X_i)$

$X_i = V_i$; $trial(i)$=0

else $trial(i) = trial(i)$+1

Step 2.6.3 Set *t*=*t*+1;

Set *i*=*i*+1, if *i*>*SN*, set *i*=1;

Step 2.7 The scout bee phase:

if *trial(i)*>*limit*, perform **Algorithm 5** to implement chaotic local search;

Step 2.8 Memorize the best solution achieved;

Step 2.9 Update *iter*=*iter*+1, if *iter*>*MCN*, return the best solution; otherwise, go to step 2.4;

Step 2.10 Achieve the input weights and hidden biases from the best solution, then the output weights are analytically calculated;

Step 3. Testing phase:

Ten-fold CV is used to predict testing samples, and return classification accuracy from the optimized model.

5 Experimental results and discussion

In this section, the effectiveness of the proposed algorithm is empirically studied on real-world data sets, and its performance is compared with six-related algorithms for classification problems that are ELM, PSO-ELM [35], IPSO-ELM [35], ABC-ELM, GABC-ELM [21], and IABC-ELM [36].

The NABC-ELM model is evaluated using a wide range of data sets given in Table 1. Each data set is obtained from UCI machine learning repository [37]. For these data sets, the number of samples ranges up to 2310, the number of features ranges up to 22, the number of classes ranges from 2 to 7. All of them are widely used for evaluating learning algorithms, because they are from real-world applications, e.g., object recognition, disease diagnosis, and image segmentation.

Table 1 Description of the data sets used for classification

Datasets	#Samples	#Features	#Classes
Thyroid	215	5	3
Wine	178	13	3
Diabetes	768	8	2
Image segmentation	2310	19	7
Iris	150	4	3
Liver disorder	345	6	2
Parkinson	195	22	2
Hepatitis	155	19	2
Breast cancer	699	9	2
Vehicle	846	18	4

To avoid feature values in greater numerical ranges dominating those in smaller numerical ranges, normalization is employed before classification. In this study, feature values are normalized into the range of $[-1, 1]$ according to Eq. (19), where x is the original value, x' is the normalized value, a and b are -1 and 1, min_j is the minimum of feature j, and max_j is the maximum of feature j:

$$x' = a + (b - a) \times \left(\frac{x - \min_j}{\max_j - \min_j} \right). \tag{19}$$

For the NABC-ELM, the number of population size SN is taken as 40, the maximum number of cycles *MCN* is set to 150, the control parameter *limit* equals 15, and the toleration rate λ is set to 0.04. The activation function is sigmoid function: $g(x) = 1/(1 + \exp(-x))$. For a fair comparison, parameter settings of related algorithms such as SN and MCN are the same as the NABC-ELM. C_1 and C_2 called learning factors of PSO are equal to constant 2. ω called inertial factor is calculated according to the formula as follows: $\omega = \omega_{\max} - iter \times (\omega_{\max} - \omega_{\min})/\text{MCN}$, where *iter* is the number of current iteration, $\omega_{\max}$ and $\omega_{\min}$ are set 0.9 and 0.4, respectively.

To gain an unbiased estimate, tenfold CV is used to evaluate the classification accuracy. The advantage of this method is that all of the test sets are independent and the reliability of the results can be improved. Due to the arbitrariness partition of data sets, tenfold CV will be repeated and averaged over 5 runs for accurate evaluation.

5.1 Comparison with related algorithms

To state the NABC-ELM model validity, performance of all the classifiers is evaluated by different measures like training accuracy, testing accuracy, the number of hidden nodes, and training time. The number of hidden nodes is varying from 5 to 40 with an increment of five nodes each time. The comparative results are presented in Table 2.

It is clearly seen from Table 2 that intelligent optimization algorithms can achieve better test accuracy with less number of hidden nodes on the majority of data sets compared.

Specially, NABC-ELM model achieves better classification performance and requires less number of hidden nodes than other models. It suggests that the parameters optimization can be effectively implemented, and a more compact SLFNs structure can be obtained from the NABC-ELM model. Meanwhile, the gap between training accuracy and test accuracy is smaller than other models. In addition, it is also clearly seen from training time that ELM model possesses much faster learning speed owing to parameters unadjusted, but classification accuracy is not good. Obviously, the intelligent optimization algorithms need an optimizing process, so they take more training time. Training time of the NABC-ELM model has little difference compared with PSO-ELM, IPSO-ELM, ABC-ELM, GABC-ELM, and IABC-ELM model, but the NABC-ELM model can achieve better classification accuracy. Therefore, based on the above analysis, it can be concluded that the proposed NABC-ELM model is able to achieve more robust performance relative to all the other ELM variants considered.

5.2 Effect of the number of hidden nodes

It is well known that performance of ELM is mainly influenced by the number of hidden nodes. Here, the key factor will be examined in detail. To investigate the impact of the factor, classification accuracy is analyzed using different numbers of hidden nodes which ranges from 5 to 40. The relationship between classification accuracy and the number of hidden nodes on six different data sets is shown in Fig. 1. The six data sets are Thyroid, Wine, Diabetes, Image segmentation, Iris, and Liver disorder, respectively. The basis for selecting them is that they are representative.

As can be seen from Fig. 1, the NABC applied to optimize input weights and biases of ELM can enhance convergence ability, which brings about better generalization performance of ELM. The number of hidden nodes has a big impact on the performance of ELM classifier. The best testing accuracy of 0.964, 0.9899, 0.7619, 0.8773, 0.9769, and 0.7322 is achieved with the number of hidden nodes 10, 10, 10, 10, 5, and 10, as shown in Fig. 1a–f.

Performance of all the algorithms on Thyroid data sets is shown in Fig. 1a. It is clearly seen that the curve of NABC-ELM is relatively flat, and the best testing accuracy is obtained when the number of hidden nodes is equal to 10. Compared with NABC-ELM, classification result of ELM is very poor when the number of hidden nodes is less than 10. Compared with other algorithms, testing accuracy of NABC-ELM is more than 0.95 even if the number of hidden nodes is less than 10. Similar situations can also be seen on other data sets.

Performance of all the algorithms on Wine data sets is shown in Fig. 1b. As can be seen from Fig. 1b, the best testing accuracy of NABC-ELM, GABC-ELM and IABC-ELM is almost the same when the number of hidden nodes is equal to 10. However, NABC-ELM and IABC-ELM are relatively stable. Performance of all the algorithms on Iris data sets is shown in Fig. 1e. When the number of hidden nodes is equal to 5, the best testing accuracy of NABC-ELM, GABC-ELM, and IABC-ELM is obtained, respectively. Meanwhile, that of GABC-ELM is higher than IABC-ELM. Performance of all the algorithms on Liver disorder data sets is shown in Fig. 1f. Testing accuracy of ELM, PSO-ELM, IPSO-ELM, and ABC-ELM decreased greatly due to overfitting when the number of hidden nodes is more than 25. In summary, it can be observed that NABC-ELM model has better stability and gives better

Table 2 Comparison of different ELM models

Datasets	Algorithm	Training accuracy	Testing accuracy	#Hiddennodes	Training time	Datasets	Algorithm	Training accuracy	Testing accuracy	#Hidden nodes	Training time
Thyroid	ELM	0.9548	0.9383	30	0.0023	Liver disorder	ELM	0.7672	0.7155	30	0.0065
	PSO-ELM	0.9653	0.9472	30	5.016		PSO-ELM	0.7738	0.7179	25	11.637
	IPSO-ELM	0.971	0.9481	25	4.825		IPSO-ELM	0.7761	0.7199	25	11.524
	ABC-ELM	0.9758	0.9593	15	4.419		ABC-ELM	0.7816	0.7289	15	11.329
	GABC-ELM	0.9789	0.9608	10	4.413		GABC-ELM	0.7822	0.7301	15	11.261
	IABC-ELM	0.9793	0.9621	10	4.408		IABC-ELM	0.7825	0.7305	10	11.229
	NABC-ELM	0.9805	0.964	10	4.411		NABC-ELM	0.7839	0.7322	10	11.184
Wine	ELM	0.9985	0.9768	25	0.0031	Parkinson	ELM	0.9236	0.8629	40	0.0065
	PSO-ELM	0.9996	0.9748	15	7.296		PSO-ELM	0.9372	0.8768	30	11.469
	IPSO-ELM	0.9996	0.9765	15	7.153		IPSO-ELM	0.9396	0.8815	25	11.412
	ABC-ELM	0.9991	0.9801	10	6.992		ABC-ELM	0.9519	0.8921	15	11.266
	GABC-ELM	0.9996	0.9895	10	6.981		GABC-ELM	0.9642	0.924	15	11.235
	IABC-ELM	0.9997	0.9899	10	6.988		IABC-ELM	0.9611	0.9183	15	11.248
	NABC-ELM	0.9996	0.9895	10	7.008		NABC-ELM	0.9685	0.9361	15	11.221
Diabetes	ELM	0.7782	0.7446	40	0.0088	Hepatitis	ELM	0.8682	0.8247	30	0.0036
	PSO-ELM	0.7841	0.7489	35	15.531		PSO-ELM	0.8716	0.8335	25	7.452
	IPSO-ELM	0.7863	0.7512	25	15.283		IPSO-ELM	0.8749	0.8376	25	7.316
	ABC-ELM	0.7899	0.7563	15	15.058		ABC-ELM	0.8805	0.8416	20	7.105
	GABC-ELM	0.7906	0.7597	15	15.035		GABC-ELM	0.887	0.8525	15	6.998
	IABC-ELM	0.7911	0.7612	15	15.011		IABC-ELM	0.8878	0.8531	15	6.991
	NABC-ELM	0.7914	0.7619	10	14.996		NABC-ELM	0.8922	0.8564	10	6.982
Image segmentation	ELM	0.7916	0.8223	40	0.0215	Breast cancer	ELM	0.9718	0.9592	40	0.0058
	PSO-ELM	0.8574	0.8441	30	36.473		PSO-ELM	0.9689	0.9602	35	9.607
	IPSO-ELM	0.8697	0.8539	25	35.629		IPSO-ELM	0.9784	0.9685	25	9.326
	ABC-ELM	0.8775	0.8644	15	36.058		ABC-ELM	0.9818	0.9662	15	9.083
	GABC-ELM	0.8931	0.8706	10	35.611		GABC-ELM	0.9852	0.9691	15	9.041
	IABC-ELM	0.8974	0.8732	10	35.620		IABC-ELM	0.9869	0.9702	10	8.926
	NABC-ELM	0.907	0.8773	10	35.622		NABC-ELM	0.986	0.9696	10	8.952
Iris	ELM	0.96	0.9542	20	0.0016	Vehicle	ELM	0.8951	0.8518	35	0.0106
	PSO-ELM	0.9638	0.9589	15	3.885		PSO-ELM	0.9025	0.8639	30	20.464
	IPSO-ELM	0.9676	0.9613	10	3.264		IPSO-ELM	0.9042	0.8681	25	19.682
	ABC-ELM	0.9763	0.9668	10	3.116		ABC-ELM	0.9126	0.8723	25	19.458
	GABC-ELM	0.9881	0.9742	5	2.905		GABC-ELM	0.9163	0.8812	20	18.712
	IABC-ELM	0.9874	0.9725	5	2.919		IABC-ELM	0.9172	0.8839	20	18.449
	NABC-ELM	0.9891	0.9769	5	2.879		NABC-ELM	0.9216	0.8861	15	17.926

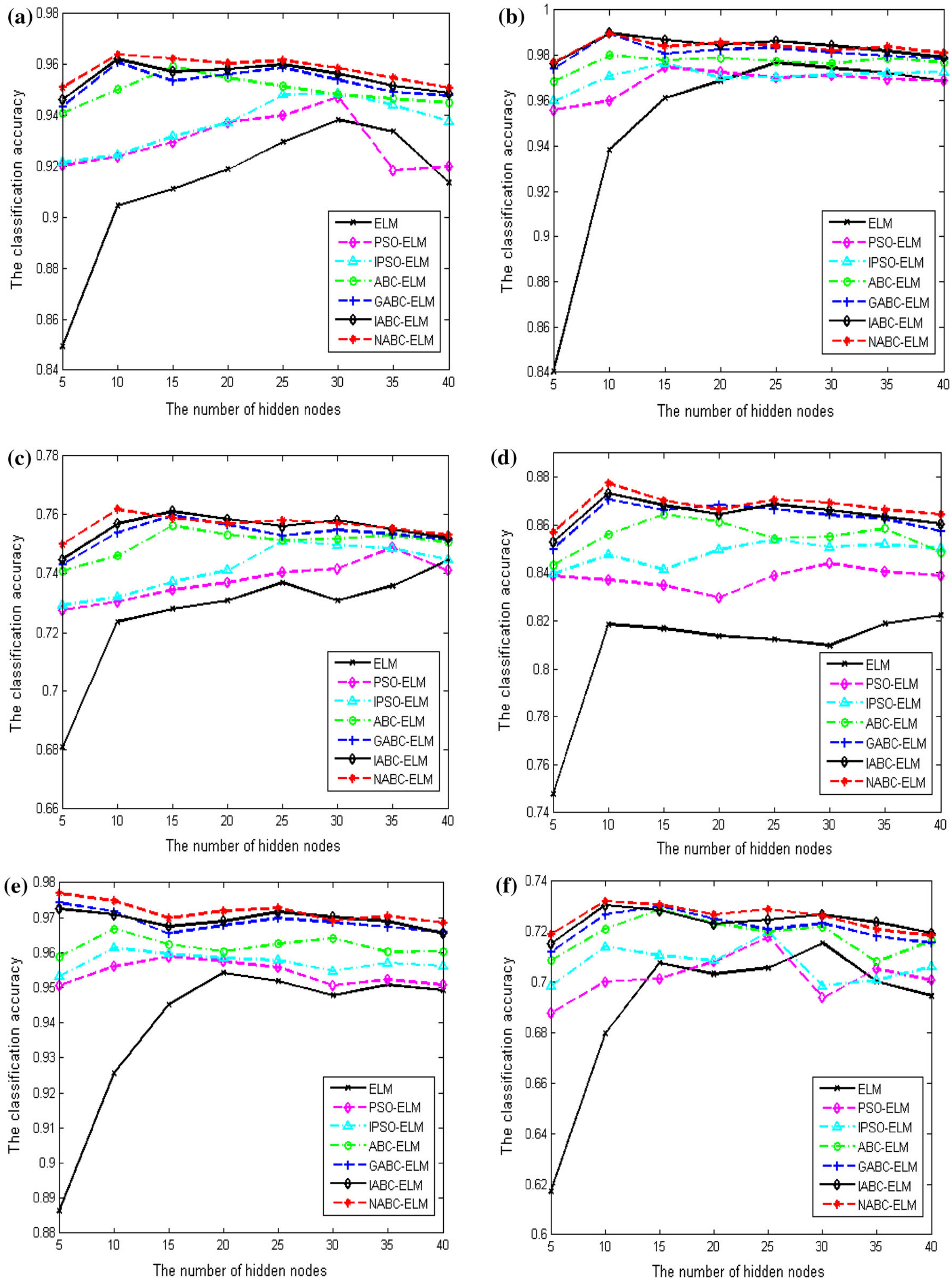

Fig. 1 The classification accuracy of the seven algorithms with different numbers of hidden nodes **a** Thyroid; **b** Wine; **c** Diabetes; **d** Image segmentation; **e** Iris; **f** Liver disorder

Table 3 Wilcoxon sign rank results obtained by NABC-ELM against other six algorithms for ten data sets

Algorithm	p value
NABC-ELM to ELM	0.028
NABC-ELM to POS-ELM	0.031
NABC-ELM to IPSO-ELM	0.036
NABC-ELM to ABC-ELM	0.039
NABC-ELM to GABC-ELM	0.043
NABC-ELM to IABC-ELM	0.046

testing accuracy with respect to the number of hidden nodes. In addition, these results demonstrate the proposed NABC-ELM model reaches a superior performance under conditions where there is a relatively small number of hidden nodes.

Furthermore, statistical testing is a meaningful way to study the difference between any two stochastic algorithms. In this paper, a non-parametric test, Wilcoxon signed rank test, is chosen to judge the difference. The statistical results based on the best testing accuracy are presented in Table 3. As a null hypothesis, there is no significant difference between the two algorithms, whereas the alternative hypothesis is assumed that there is a significant difference between the two algorithms at the 5 % significance level. The well-known statistical software packages SPSS 19 is used for the computation of the p value for these tests. From Table 3, NABC-ELM is clearly superior to some existing ELM variants.

6 Conclusions

A novel artificial bee colony algorithm for optimization problems, called NABC, is proposed in this paper. In NABC, chaotic opposition-based learning initialization method is employed to improve the quality of initial population. The self-adaptive search strategy is presented to enhance convergence ability. The chaotic local search for scout bee is implemented to further help to escape from local optima. Due to these advantages, the NABC is applied to optimize input weights and biases of ELM so as to improve generalization performance of ELM. As a consequence, NABC-ELM considered as a very efficient and robust algorithm is presented. To validate the proposed model, it is conducted on real-world data sets for classification problems. Experimental results show that NABC-ELM outperforms other compared algorithms, and may be a promising and viable tool to deal with classification problems.

References

1. Jain, A.K., Mao, J.: Artificial neural networks: a tutorial. IEEE Comput. **29**, 31–44 (1996)
2. Siniscalchi, S.M., Svendsen, T., Lee, C.H.: An artificial neural network approach to automatic speech processing. Neurocomputing **140**, 326–338 (2014)
3. Dai, Q., Liu, N.: Alleviating the problem of local minima in backpropagation through competetive learning. Neurocomputing **94**, 152–158 (2012)
4. Cortes, C., Vapnik, V.N.: Support vector networks. Mach. Learn. **20**, 273–297 (1995)
5. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
6. Huang, G.B., Ding, X., Zhou, H.: Optimization method based extreme learning machine for classification. Neurocomputing **74**, 155–163 (2010)
7. Li, L.N., et al.: A computer aided diagnosis system for thyroid disease using extreme learning machine. J. Med. Syst. **36**(5), 3327–3337 (2012)
8. Zhao, X., Wang, G., Bi, X., Gong, P., Zhao, Y.: XML document classification based on elm. Neurocomputing **74**(16), 2444–2451 (2011)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE Int. Conf. Neural Networks, pp. 1942–1948 (1995)
10. Dorigo, M., Stutzle, T.: Ant colony optimization. MIT Press, Cambridge, MA (2004)
11. Tang, K.S., Man, K.F., Kwong, S., He, Q.: Genetic algorithms and their applications. IEEE Signal Process. Mag. **13**, 22–37 (1996)
12. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
13. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**, 687–697 (2008)
14. Karaboga, D., Basturk, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**, 108–132 (2009)
15. Huang, Y.M., Lin, J.C.: A new bee colony optimization algorithm with idle-time-based filtering scheme for open shop-scheduling problems. Expert Syst. Appl. **38**, 5438–5447 (2011)
16. Karaboga, D., Ozturk, C., Karaboga, N., Gorkemli, B.: Artificial bee colony programming for symbolic regression. Inf. Sci. **209**, 1–15 (2012)
17. Shayeghi, H., Ghasemi, A.: A modified artificial bee colony based on chaos theory for solving non-convex emission/economic dispatch. Energy Convers. Manage **79**, 344–354 (2014)
18. Singh, A., Sundar, S.: An artificial bee colony algorithm for the minimum routing cost spanning tree problem. Soft Comput. **15**, 2489–2499 (2011)
19. Xiang, W.L., An, M.Q.: An efficient and robust artificial bee colony algorithm for numerical optimization. Comput. Oper. Res. **40**, 1256–1265 (2013)
20. Basturk, B., Karaboga, D.: A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. **192**, 120–142 (2012)
21. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**, 3166–3173 (2010)
22. Gao, W.F., Liu, S.Y.: Improved artificial bee colony algorithm for global optimization. Inf. Process. Lett. **111**, 871–882 (2011)
23. Li, G., Niu, P., Xiao, X.: Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Appl. Soft Comput. **12**, 320–332 (2012)
24. Yurtkuran, A., Emel, E.: An adaptive artificial bee colony algorithm for global optimization. Appl. Math. Comput. **271**, 1004–1023 (2015)
25. Maeda, M., Tsuda, S.: Reduction of artificial bee colony algorithm for global optimization. Neurocomputing **148**, 70–74 (2015)
26. Gao, W.F., Huang, L.L., et al.: Artificial bee colony algorithm with multiple search strategies. Appl. Math. Comput. **271**, 269–287 (2015)

27. Shan, L., Qiang, H., Li, J., et al.: Chaotic optimization algorithm based on Tent map. Control Decis. **20**(2), 179–182 (2005)
28. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition versus randomness in soft computing techniques. Appl. Soft Comput. **8**, 906–918 (2008)
29. Alatas, B.: Chaotic bee colony algorithms for global numerical optimization. Expert Syst. Appl. **37**, 5682–5687 (2010)
30. Gao, W.F., Liu, S.Y.: A modified artificial bee colony algorithm. Comput. Oper. Res. **39**, 687–697 (2012)
31. Liao, X., Zhou, J.Z., et al.: An adaptive chaotic artificial bee colony algorithm for short-term hydrothermal generation scheduling. Electr. Power Energy Syst. **53**, 34–42 (2013)
32. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: The best-so-far selection in artificial bee colony algorithm. Appl. Soft Comput. **11**, 2888–2901 (2011)
33. Gao, W.F., Liu, S.Y., Huang, L.L.: Enhancing artificial bee colony algorithm using more information-based search equations. Inf. Sci. **270**, 112–133 (2014)
34. Bartlett, P.L.: The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans. Inf. Theory **44**(2), 525–536 (1998)
35. Han, F., Yao, H.F., Ling, Q.H.: An improved evolutionary extreme learning machine based on particle swarm optimization. Neurocomputing **116**, 87–93 (2013)
36. Gao, W.F., Liu, S.Y., Huang, L.L.: Inspired artificial bee colony algorithm for global optimization problems. Acta Electron. Sin. **40**(12), 2396–2403 (2012)
37. University of California, Irvine, Machine Learning Repository. http://archive.ics.uci.edu/ml/