Mathematical Biosciences and Engineering

MBE, 16(5): 4692–4707.
DOI: 10.3934/mbe.2019235
Received: 21 February 2019
Accepted: 13 May 2019
Published: 23 May 2019

http://www.aimspress.com/journal/MBE

Research article

Tuning extreme learning machine by an improved electromagnetism-like mechanism algorithm for classification problem

Mengya Zhang, Qing Wu* and Zezhou Xu

College of Engineering, Huazhong Agricultural University, Wuhan, Hubei, 430070, China

*** Correspondence:** Email: wuqing@mail.hzau.edu.cn; Tel: +861365983986.

Abstract: Extreme learning machine (ELM) is a kind of learning algorithm for single hidden-layer feedforward neural network (SLFN). Compared with traditional gradient-based neural network learning algorithms, ELM has the advantages of fast learning speed, good generalization performance and easy implementation. But due to the random determination of input weights and hidden biases, ELM demands more hidden neurons and cannot guarantee the optimal network structure. Here, we report a new learning algorithm to overcome the disadvantages of ELM by tuning the input weights and hidden biases through an improved electromagnetism-like mechanism (EM) algorithm called DAEM and Moore-Penrose (MP) generalized inverse to analytically determine the output weights of ELM. In DAEM, three different solution updating strategies inspired by dragonfly algorithm (DA) are implemented. Experimental results indicate that the proposed algorithm DAEM-ELM has better generalization performance than traditional ELM and other evolutionary ELMs.

Keywords: extreme learning machine; electromagnetism-like mechanism; dragonfly algorithm; classification problem

1. Introduction

Classification is a very important issue in many fields such as face detection, big data, and disease diagnosis. Especially in recent years, with the development of internet and smart devices, various types of data are exploding. In order to obtain accurate results more efficiently, the traditional image analysis method and signal detection are being replaced by machine learning methods gradually. In all these methods, artificial neural networks (ANNs) [1] and support vector machine

(SVM) [2] are the two most popular methods. For ANNs, many neural network models have been developed, such as back propagation algorithm (BP) [3] and convolutional neural networks (CNN) [4]. However, these techniques are time-consuming, easy to be trapped in local optima and require the setting of many parameters. To overcome these disadvantages, the extreme learning machine (ELM) method has been proposed for single-hidden layer feed-forward neural network [5].

ELM has advantages of high learning speed and excellent classification performance owing to its inherent characteristics of simple structure. Due to the above advantages, ELM has been widely used in various fields, such as localization [6], industrial production [7], solar radiation prediction [8], finite-time optimal control of nonlinear systems [9], etc. In addition, ELM has several variants to solve complex problems. Zhang et al. proposed a multilayer probability extreme learning machine for device-free localization [10]. Youngmin Park combined convolutional neural network and ELM for image classification [11]. In ELM, the input weights and hidden bias are randomly generated without iterative learning [12]. Although these settings bring certain advantages, they also increase the risk of overfitting. Besides, the hidden neurons are sensitive to unknown testing data. Traditionally, choice of these parameters mainly depends on prior knowledge and expertise. To solve these problems, it is important to optimize the input weights, hidden bias and the structure of the network.

Intelligent algorithms are naturally considered as the solution to the above problems, such as particle swarm optimization (PSO) [13], ant colony optimization (ACO) [14], and artificial bee colony algorithm (ABC) [15]. Electromagnetism-like mechanism (EM) algorithm, which was developed by Birbil and Fang in 2003 [16], is a population-based random search algorithm similar to genetic algorithm (GA). Because of its strong search capability and easy implementation, EM has been successfully applied to optimization problems [16–23], such as function optimization [17] and flow shop scheduling [18–21]. All these previous studies have demonstrated the excellent optimization performance of EM. Therefore, the integration of EM and ELM should be a promising approach in training feedforward neural network.

In this study, an improved EM algorithm called DAEM is proposed by incorporating some theories of dragonfly algorithm (DA) [24] into EM approach. By using the new algorithm, we optimized the input weights and hidden biases, and minimum norm least-square scheme was used to analytically determine the output weights in ELM. In the selection of input weights and hidden biases, the improved EM considers not only the classification error rate but also the norm of the output weights as well as constrains the input weights and hidden biases within a reasonable range. In addition, the *k*-fold cross-validation method is adopted to avoid the problem of overfitting.

The rest of the paper is organized as follows. The theories related to ELM and EM are briefly introduced in Section 2. Section 3 describes the establishment of the DAEM-ELM algorithm. Section 4 presents the results and discussion on eight classification problems to demonstrate the effectiveness of the proposed algorithm. Finally, the conclusions are summarized in Section 5.

2. Theories and methods

2.1. Extreme learning machine

The core idea of ELM is to transform the training process of traditional SLFN model into solving the least square solution problem. The main process of ELM consists of random generation of the

parameters of hidden neurons, followed by fixing of the hidden layer parameters and then algebraically solving the output weights. The specific theoretical basis of ELM is as follows.

For N arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$, $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$. The i-th sample x_i is an $n \times 1$ feature vector, and t_i is an $m \times 1$ target vector. The standard mathematical model of SLFNs with L hidden neurons and activation function $g(x)$ is as follows:

$$O_j = \sum_{i=1}^{L} \beta_i g(\omega_i \cdot x_j + b_i), j = 1, 2, ..., N \tag{1}$$

where O_j denotes the corresponding actual output vector of x_j, $\omega_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{in}]^T$ indicates the weight vector connecting the i-th hidden neuron and input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ represents the weight vector connecting the i-th hidden neuron and output neurons, b_i is the bias of i-th hidden neuron, also known as the threshold, and $\omega_i \cdot x_j$ is the inner product of ω_i and x_j. The purpose of training SLFNs is to minimize the error of output value， which means:

$$\min \sum_{j=1}^{L} \left\| O_j - t_j \right\| \tag{2}$$

Then, the N equations represented by equation (1) can be expressed in matrix form as follows:

$$H\beta = T \tag{3}$$

where $H = H(\omega_1, \omega_2, ..., \omega_L, b_1, b_2, ..., b_L, x_1, x_2, ..., x_N) = \begin{bmatrix} g(\omega_1 \cdot x_1 + b_1) & \ldots & g(\omega_1 \cdot x_1 + b_L) \\ \ldots & \ldots & \ldots \\ g(\omega_1 \cdot x_N + b_1) & \ldots & g(\omega_L \cdot x_N + b_L) \end{bmatrix}_{N \times L}$,

$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}$, $T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m}$, H is the hidden layer output matrix, β is the output weight matrix, and T is the output matrix.

In the algorithm of ELM, when the input weights and hidden layer biases are randomly generated, the determination of the output weights is to find the least-square (LS) solution to the linear system:

$$\beta = H^{+}T \tag{4}$$

where H^{+} is obtained by singular value decomposition of Moore-Penrose (MP) generalized inverse matrix.

The pseudo-code of ELM is as follows:

Input: (x_i, t_i), L, $g(x)$

Generate the input weights ω_i and the biases b_i of hidden neurons randomly;

$H \leftarrow$ Compute the hidden layer output matrix;

$\beta \leftarrow$ Compute the output weights by formula (4);

Output: β

2.2. Electromagnetism-like mechanism algorithm

The basic principle of EM is that every feasible solution is compared to a charged particle and the charge of each particle is calculated by the value of the preparative optimization objective function [16]. The charge determines not only the type of the force between two particles (either attraction or repulsion), but also the strength of the force. Under the action of attraction and repulsion forces, the population moves to a new generation. The whole process of particle movement under the force in a population is shown in Figure 1. As can be clearly seen from Figure 1, the blue particle is subject to the forces of other particles in the population, both attraction and repulsion. Besides, the optimal particle in the population will always attract other particles to move towards it. Specifically, the EM algorithm mainly includes the following four steps.

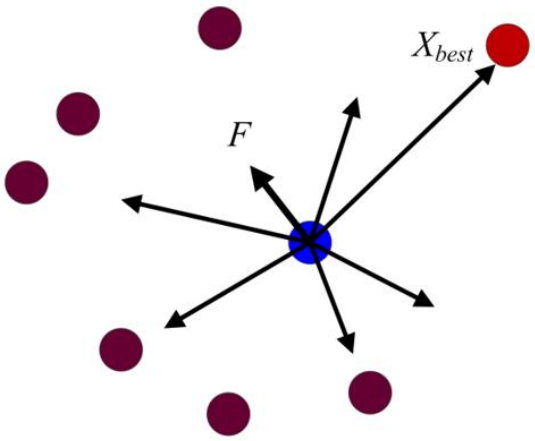

Figure 1. The process of particle movement.

(1) **Initialization**: *m* particles are randomly selected from the feasible region as initial population. Each coordinate of the particle is uniformly distributed between corresponding upper and lower bounds. Then, the objective function value is calculated for each particle and the particle with the best objective function value is stored in X_{best}.

(2) **Local search**: The procedure of the local search conducted on a single particle is to improve the solution obtained. Each dimension of the current optimal particle X_{best} is searched according to a certain step size. Once a better solution is found, the optimal particle is updated. The effective local information obtained from this procedure can contribute to the abilities of EM of both exploration and exploitation.

(3) **Calculation of the resultant force**: The magnitude of the force of particle *i* is strongly related to its charge q_i, which can be calculated by formula (5):

$$q_i = \exp\left(-n\frac{f(X_i)-f(X_{best})}{\sum_{k=1}^{m}\left(f(X_k)-f(X_{best})\right)}\right), \forall i \tag{5}$$

The resultant force F_i exerted on particle *i* can be calculated as follows:

$$F_i = \sum_{j \neq i}^{m} \begin{cases} \left(X_j - X_i\right) \dfrac{q_i q_j}{\left\|X_j - X_i\right\|^2}, \ if \ f\left(X_j\right) < f\left(X_i\right) \\ \left(X_i - X_j\right) \dfrac{q_i q_j}{\left\|X_j - X_i\right\|^2}, \ if \ \ f\left(X_j\right) \geq f\left(X_i\right) \end{cases}, \forall i \tag{6}$$

According to the above formula, the particle with better and poorer objective function values attracts and repulses other particles, respectively. The better the objective function value is, the stronger the attraction will be, and vice versa.

(4) **Movement of the population**: After calculating the resultant force, the particles are moved in the direction of the force, thus forming a new generation of population. The direction and step of the movement are determined by the following formula:

$$X_i = X_i + \lambda \frac{F_i}{\left\|F_i\right\|}\left(RNG\right), \forall i \tag{7}$$

λ is a random number of 0 to 1, which guarantees that the particles with a nonzero probability move to unvisited regions. *RNG* is a movable range between the upper and lower sectors.

The pseudo-code of EM algorithm is as follows:

EM (*M*, *MAXITER*, *LSITER*, *δ*)

M is the number of particles; ***MAXITER*** is the maximum number of iterations; ***LSITER*** is the maximum number of iterations in local search; ***δ*** is the local search parameter, $\delta \in [0,1]$.

```
Initialization of the population and parameters
iteration ← 0
Do {
Local search (LSITER, δ)
q ← Calculation of charge of each particle (f(X))
F ← Calculation of resultant force (q)
Move each particle (F)
iteration ← iteration + 1
} while iteration < MAXITER
```

3. The improved extreme learning machine (DAEM-ELM)

From the introduction of ELM, it can be seen that the input weights and hidden layer biases are randomly generated at the initialization stage. The network constructed in this way may give rise to a problem of overfitting. More specifically, ELM usually requires more hidden neurons than conventional neural networks to achieve the expected performance. Larger network size results in longer running time of the testing phase of ELM, which may hinder its efficient development in some test time sensitive scenarios [25]. To solve this problem, an improved approach designated as DAEM-ELM, which combines EM with ELM, is proposed in this paper. This new ELM adopts a novel EM called DAEM to optimize the input weights and biases of ELM to improve the generalization performance and the conditioning of the SLFN. In this section, we will first provide a

detailed description of the DAEM algorithm, and then present the DAEM-ELM algorithm.

3.1. DAEM

The convergence speed of EM slows down gradually during the iterations and the algorithm easily falls into the local optimal solution, that is, prematurity. In addition, the position and number of adjacent particles influence the step length and position update of the population according to formula (1). But it remains unclear under what conditions individual particles can be defined as adjacent to each other. In order to solve these problems, we propose an improved EM algorithm called DAEM, by incorporating some theories of dragonfly algorithm (DA) into EM.

First, the adjacency of particles is defined. A neighborhood (circle in a 2D, sphere in a 3D space, or hypersphere in an *n*D space) with a certain radius *r* is assumed around each particle [24]. If the Euclidean distance between particle *i* and particle *j* is less than *r*, particle *i* and particle *j* are considered as adjacent. In order to accelerate the convergence speed, the radius *r* increases with increasing number of iterations. The specific calculation formula of *r* is as follows:

$$r = \frac{ub - lb}{4} + 2 \times \frac{iter \times (ub - lb)}{Max\_iteration} \tag{8}$$

where *iter* is the number of current iterations, *Max_iteration* is the maximum number of iterations, *ub* is the upper limit of variables, and *lb* is the lower limit of variables. The neighborhood can be represented as (r_1, r_2,..., r_n) and *n* represents the number of dimensions.

Taking particle *i* as an example, it can be expressed as X_i = (x_{i1},x_{i2},..., x_{in}), and then the neighborhood range of particle *i* in the *j*-th dimension (j = 1, 2,…, n) is [$x_{ij} - r_j$, $x_{ij} + r_j$]. When another particle X_k is within the neighborhood range of particle *i* of each dimension, it is considered that X_k is adjacent to X_i.

For different problems, the fixed solution updating strategy of EM may not be always reasonable, and cannot guarantee the discovery of global optimal solution or approximate global optimal solution. Therefore, DAEM provides three different solution updating strategies motivated by DA. In this way, the suitable updating strategy can be selected according to the prior information of different problems. Besides, variable searching step is adopted to solve the conflicts between solution accuracy and computation time in the optimization process. The three updating strategies are described as follows.

Strategy 1: when the distance between the current particle and the optimal particle in a certain dimension is smaller than the neighborhood radius, and there are adjacent particles in the neighborhood of the current particle, the updated formula of the particle is as follows:

$$X_i = X_i + c \times \eta \times \frac{F_i}{\|F_i\|} \tag{9}$$

where c = 0.9–*iter* × (0.5/*Max_iteration*)， η is a random number between [0,1]. F_i is still calculated by formula (6), but only the particles in the neighborhood instead of the entire population are considered.

Strategy 2: when the distance between the current particle and the optimal particle in a certain dimension is smaller than the neighborhood radius, and there is no particle adjacent to the current particle in its neighborhood, the updated formula of the particle is as follows:

$$X_i = X_i + levy\left(X_i\right) \times X_i \tag{10}$$

$$levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{1/\alpha}} \tag{11}$$

$$\sigma = \left(\frac{\Gamma(1+\alpha) \times \sin\left(\frac{\pi\alpha}{2}\right)}{\Gamma\left(\frac{1+\alpha}{2}\right) \times \alpha \times 2^{\left(\frac{\alpha-1}{2}\right)}} \right)^{1/\alpha} \tag{12}$$

$$\Gamma(x) = (x-1)! \tag{13}$$

where r_1 and r_2 are two random numbers in [0,1], and α is a constant (equal to 1.5 in DAEM).

Strategy 3: when the distance between the current particle and the optimal particle in any dimension is greater than the neighborhood radius, the updated formula of the particle is as follows:

$$X_i = X_i + c \times \Delta X_i + \varepsilon_W \times Worst + \varepsilon_B \times Best \tag{14}$$

$$\Delta X_i = l * L + e * E + c * C \tag{15}$$

where ε_W and ε_B are random numbers in [0,1], $Worst = X_{worst} + X_i$, $Best = X_{best} - X_i$, X_{worst} and X_{best} represent the worst and best particle of current population respectively, l is a random numbers in [0,1], L is the difference between the position of X_{best} and position of X_i when X_{best} is in the neighborhood of X_i, otherwise L is a m-dimensional vector of 0's, $e = 0.1 - iter \times (0.2/Max\_iteration)$ (when $e < 0$, let $e = 0$), C is an n-dimensional vector generated randomly.

3.2. DAEM-ELM

The basic process of DAEM-ELM is described below. Data samples are divided into training sample set and testing sample set. The training set is trained by DAEM to build the classification model. Then, the testing set is tested by the obtained classification model. The detailed steps of the proposed method are as follows.

Firstly, the population is randomly generated. Each particle in the population is composed of a set of input weights and hidden biases. The specific coding form is as follows:

$$X_i = \{\omega_{11}, \omega_{12}, ..., \omega_{1L}, \omega_{21}, \omega_{22}, ..., \omega_{2L}, ..., \omega_{n1}, \omega_{n2}, ..., \omega_{nL}, b_1, b_2, ..., b_L\} \tag{16}$$

where N is the number of input neurons, and L is the number of hidden neurons, that is, the dimension of each particle is $(N + 1) \times L$. All components in the particle are randomly initialized within the range of [–1, 1].

Secondly, for each particle, the corresponding output weights are computed according to formula (4). Then, the fitness of each particle is evaluated. The fitness function formula of DAEM-ELM is:

$$f = \frac{\sum_{i=1}^{N} MCR_i}{N} \quad i = 1, 2, ..., N \tag{17}$$

where N is the number of training samples, and MCR_i is the misclassification of the algorithm.

Neural network training should not solely rely on the misclassification of training set as the fitness function, because a higher training accuracy does not guarantee a higher test accuracy. It has been reported that neural networks tend to have better generalization performance with weights of

smaller norms [26,27]. Hence, the output weights are also considered in selection strategy design in order to further enhance the performance of our algorithm. It is stipulated that when the fitness values of different particles are similar, the particle with smaller norm of output weights is chosen as a better solution. The algorithm introduces the tolerance rate λ to meet this requirement and the effect is better when λ is set to 0.04 as validated by experiment. Besides, to reduce the computational complexity of the algorithm, the following regulation formula of current optimal solution is used only after each iteration (the fitness function is still the misclassification, and only the tolerance rate is added for adjustment).

$$X_{ibest} = \begin{cases} X_i & if\left(f\left(X_{ibest}\right) - f\left(X_i\right) > \lambda f\left(X_{ibest}\right)\right) or\left(\left|f\left(X_{ibest}\right) - f\left(X_i\right)\right| < \lambda f\left(X_{ibest}\right) \& \left\|\beta\left(X_i\right)\right\| < \left\|\beta\left(X_{ibest}\right)\right\|\right) \\ X_{ibest} & else \end{cases} \tag{18}$$

$$X_{gbest} = \begin{cases} X_{ibest} & if\left(f\left(X_{gbest}\right) - f\left(X_{ibest}\right) > \lambda f\left(X_{gbest}\right)\right) or\left(\left|f\left(X_{gbest}\right) - f\left(X_{ibest}\right)\right| < \lambda f\left(X_{gbest}\right) \& \left\|\beta\left(X_{ibest}\right)\right\| < \left\|\beta\left(X_{gbest}\right)\right\|\right) \\ X_{gbest} & else \end{cases} \tag{19}$$

where $f(X_i)$, $f(X_{ibest})$ and $f(X_{gbest})$ are the corresponding fitness values for the i-th particle, the best position of the i-th particle and global best position of all particles, respectively. $\beta(X_i)$, $\beta(X_{ibest})$ and $\beta(X_{gbest})$ are the corresponding output weights obtained by MP generalized inverse when the input weights are set as the i-th particle, the best position of the i-th particle and global best position of all particles, respectively.

Thirdly, the k-fold cross-validation (k-fold CV) method is applied in order to get an unbiased estimate of the generalization accuracy and make full use of samples in case of insufficient sample. In k-fold CV, the data sample set is divided into k mutually disjoint subsets (approximately equal in size), such as $S_1, S_2, \ldots, S_k$, and then DAEM-ELM is performed for k iterations. S_k is selected as the testing set and the rest of subsets are used as training set in the i-th iteration. Here, the parameter value of k is 5 and the final classification results are the average value of five iterations.

The pseudo-code of DAEM-ELM is as follows:

DAEM-ELM:
Performance estimation by k-fold CV where k=5;
MAXITER is the maximum number of iterations.

begin
for i=1:k
Training set=k-1 subsets Testing set=remaining subsets
begin DAEM
Initialize the population with random numbers
iteration ← 0
Do{
Train the ELM on the training set
Calculate fitness value
Update the position of each particle
iteration ← *iteration* + 1
} while iteration < *MAXITER*
end DAEM
Achieve the optimal input weights and hidden bias from the best

solution
Test the ELM with the optimal input weights and hidden bias
end
Return the average classification accuracy and standard deviation of ELM
end

4. Experiment and discussion

4.1. Datasets and experimental setup

In this section, the performance of the proposed algorithm is evaluated on eight real-world classification problems (Thyroid, WDBC, Wine, Bupa Liver, Australian, Breast Cancer, Parkinson and Iris), and all these data-sets are taken from the University of California Irvine (UCI) repository [28]. The specification of these datasets is listed in Table 1.

Table 1. Specification of eight classification problems.

Data-sets	Instances	Attributes	Classes	Missing Value
Thyroid	215	5	3	N
Parkinson	195	22	2	N
Iris	150	4	3	Y
Bupa Liver	345	6	2	N
Australian	690	14	2	Y
Breast Cancer	699	9	2	Y
Wine	178	13	3	N
WDBC	569	30	2	N

Iris, Australian and Breast cancer datasets have missing values (A data-set contains a certain number of instances, and an instance includes several attributes. If some instance in a data-set lack some attributes, it is said that the data-set has missing values.). The missing categorical attributes are replaced by the mode of the attributes, and the missing continuous attributes are replaced by the mean of the attributes in order to ensure the integrity of the sample data. Besides, normalization is employed to avoid the influence of the feature values in larger numerical ranges on those in smaller numerical ranges, which can also reduce the computational complexity. Every feature value can be normalized by scaling them into the interval of [–1, 1] according to:

$$x' = 2 \times \left(\frac{x - \min_i}{\max_i - \min_i} \right) - 1 \tag{20}$$

where x' is the normalization value, x is the original value, $\min_i$ is minimum value of feature i and $\max_i$ is maximum value of feature i.

The results obtained with the proposed algorithm are then presented and analyzed, and compared with those obtained using other related algorithms. The parameters in all algorithms of experiments are determined by trial and error. For DAEM-ELM, the maximum optimization epochs are 70, and the

population size is 50. The sigmoid function $g(x) = 1/(1 + \exp(-x))$ is adopt as the ELM activation function to compute the hidden layer output matrix. All the programs are run in MATLAB 7.0 environment.

4.2. Benchmark classification

The performance of DAEM-ELM algorithm is tested with the number of hidden neurons increasing from 5 to 40 at a step size of 5. The reason for choosing this range is that an excessive number of hidden neurons will lead to an overfitting problem for ELM. In addition, better parameters more suitable for associated networks can be found with the DAEM algorithm. Hence, the ELM only needs a small number of hidden neurons to obtain better results. The experimental results are shown in Table 2.

Table 2. Accuracy with different numbers of hidden neurons.

Hidden Neurons	Accuracy (%)	Dataset							
		Thyroid	Parkinson	Iris	Bupa Liver	Australian	Breast Cancer	Wine	WDBC
5	Training	94.19	88.46	**98.31**	78.62	76.27	94.61	98.89	97.80
	Testing	93.02	82.05	**97.33**	75.36	76.09	94.38	95.94	94.83
10	Training	97.67	89.10	98.33	80.80	82.25	94.79	**99.26**	**98.01**
	Testing	93.02	87.17	92.33	71.01	74.64	92.41	**98.33**	**96.83**
15	Training	98.49	93.59	98.33	**81.88**	83.88	95.68	99.29	97.17
	Testing	94.41	87.18	91.67	**80.65**	74.64	94.41	94.52	96.56
20	Training	**98.84**	92.95	98.33	81.88	83.88	**99.04**	100	98.19
	Testing	**95.35**	89.74	86.67	68.41	73.82	**97.57**	97.25	96.73
25	Training	**98.84**	93.59	98.33	82.25	87.50	96.05	100	97.80
	Testing	**95.35**	87.18	83.33	71.01	86.96	93.82	95.94	96.73
30	Training	98.84	96.15	98.33	82.03	90.22	96.97	100	97.80
	Testing	93.02	87.18	80.00	70.72	86.96	93.82	91.77	96.73
35	Training	**98.96**	95.51	98.33	82.97	**91.30**	97.13	100	98.19
	Testing	**95.35**	86.87	90.00	69.66	**89.13**	95.08	94.52	96.56
40	Training	98.84	**96.15**	98.33	80.07	83.88	97.31	100	99.12
	Testing	90.70	**92.31**	83.33	71.01	82.61	92.97	94.38	96.56

As shown in Table 2, the accuracy rate does not simply increase with the number of hidden neurons. When the hidden neurons increase to a certain number, further increase will lead to a decline in accuracy. Secondly, the optimal number of hidden neurons varies for different problems. Specifically, the optimal number is 20, 25 and 35 for the Thyroid dataset, and we chose the smallest number (20) to reduce the computational time of the network. For the Parkinson, Iris, Bupa Liver, Australian, Breast Cancer, Wine and WDBC datasets, the optimal number is 40, 5, 15, 35, 20, 10 and 10, respectively.

Table 3 and Table 4 show the results achieved with all seven investigated methods (ABC-ELM [29], ELM, PSO-ELM [29], IPSO-ELM [29], E-ELM [29], LM and DAEM-ELM) for the eight benchmark classification datasets based on five trials. The last column of Table 3 shows the smallest number of

hidden neurons to be used in order to achieve the best results. It is evident that a higher accuracy rate and a smaller number of hidden neurons represent a better mode. In addition, to test the accuracy, the nearly optimal number of hidden neurons for these algorithms and the standard deviations are shown in the table in the form of mean ± standard deviation. From Table 3 and Table 4, we can draw the following conclusions.

Table 3. Detailed results obtained by seven investigated algorithms via 5-fold on the eight datasets.

Dataset	Algorithm	Accuracy (%)		Hidden Neurons
		Training	Testing	
Thyroid	ABC-ELM	98.79	94.97±1.44	15
	ELM	96.84	92.93±3.98	30
	PSO-ELM	97.92	94.14±3.67	30
	IPSO-ELM	98.33	94.31±2.65	25
	E-ELM	98.10	92.74±3.02	40
	LM	95.70	91.07±3.41	35
	DAEM-ELM	**98.84**	**95.35**	**20**
Parkinson	ABC-ELM	95.12	89.11±3.02	15
	ELM	92.28	86.15±5.79	40
	PSO-ELM	93.66	87.59±4.70	30
	IPSO-ELM	93.95	88.10±4.62	25
	E-ELM	94.17	87.08±5.70	35
	LM	89.24	82.38±4.65	35
	DAEM-ELM	**96.15**	**91.80±2.30**	**40**
Iris	ABC-ELM	97.63	96.68±1.83	15
	ELM	96.00	95.42±2.45	20
	PSO-ELM	96.38	95.89±1.13	15
	IPSO-ELM	96.76	96.13±1.64	10
	E-ELM	98.81	95.20±3.13	30
	LM	98.74	96.00±2.67	10
	DAEM-ELM	**98.31**	**97.33±2.67**	**5**
Bupa Liver	ABC-ELM	77.94	72.83±4.21	15
	ELM	76.58	71.30±5.14	30
	PSO-ELM	77.18	71.54±5.26	25
	IPSO-ELM	77.40	71.72±5.33	25
	E-ELM	76.26	71.19±5.70	20
	LM	74.91	69.37±6.04	35
	DAEM-ELM	**81.88**	**80.65±0.41**	**15**

For the Thyroid dataset, although the number of hidden neurons of DAEM-ELM is not the smallest (only slightly bigger than that of ABC-ELM), its classification accuracy is the highest among seven methods and the standard deviation of the acquired performance is also the smallest (equal to 0), indicating the consistency and stability of the proposed method. The results of the Thyroid dataset are shown in Figure 2(a).

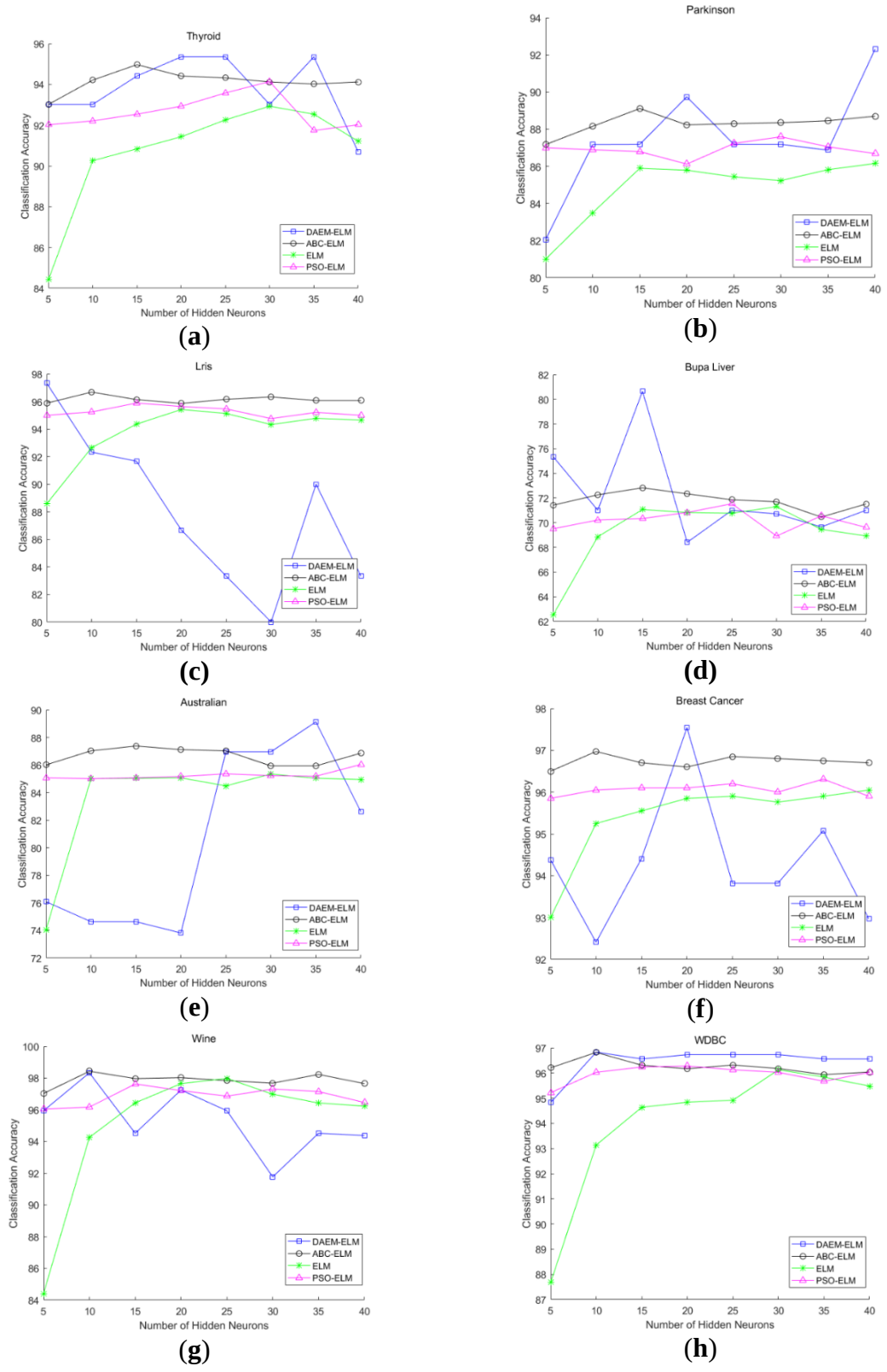

Figure 2. The classification accuracy of the four algorithms with different numbers of hidden nodes: (**a**) Thyroid; (**b**) Parkinson; (**c**) Iris; (**d**) Bupa Liver; (**e**) Australian; (**f**) Breast Cancer; (**g**) Wine; (**h**) WDBC.

Table 4. Detailed results obtained by seven investigated algorithms via 5-fold on the eight datasets.

Dataset	Algorithm	Accuracy (%)		Hidden Neurons
		Training	Testing	
Australian	ABC-ELM	90.74	87.38±1.61	15
	ELM	87.50	85.35±3.20	30
	PSO-ELM	89.37	86.04±2.31	40
	IPSO-ELM	89.65	86.41±2.72	15
	E-ELM	89.51	86.03±2.80	20
	LM	87.82	85.97±2.77	40
	DAEM-ELM	**91.30**	**89.13±0.21**	**35**
Breast Cancer	ABC-ELM	98.54	96.97±1.09	10
	ELM	97.42	96.05±1.02	40
	PSO-ELM	97.16	96.31±1.25	35
	IPSO-ELM	98.25	97.18±1.33	25
	E-ELM	97.88	96.45±1.67	35
	LM	96.21	95.96±2.24	40
	DAEM-ELM	**99.04**	**97.54±0.35**	**20**
wine	**ABC-ELM**	**99.97**	**98.43±1.11**	**10**
	ELM	99.86	97.98±2.08	25
	PSO-ELM	100	97.63±2.27	15
	IPSO-ELM	100	97.82±2.01	15
	E-ELM	100	98.02±1.69	25
	LM	99.40	98.05±2.55	30
	DAEM-ELM	99.26	98.33±1.33	10
WDBC	ABC-ELM	98.85	96.82±1.23	10
	ELM	96.43	96.13±1.64	30
	PSO-ELM	97.49	96.28±1.60	20
	IPSO-ELM	97.96	96.54±1.51	10
	E-ELM	98.03	96.10±1.93	20
	LM	96.11	95.17±2.22	30
	DAEM-ELM	**98.01**	**96.83±0.42**	**10**

For the Parkinson dataset, ABC-ELM has the fewest hidden neurons. But the proposed method outperforms other six methods in terms of classification accuracy by around 3% (more than 9% compared with that of LM). Besides, DAEM-ELM also has the smallest standard deviation among these methods. Figure 2(b) shows the accuracy obtained on the Parkinson dataset.

For the Iris dataset, DAEM-ELM only needs five hidden neurons to achieve the highest classification accuracy. In this way, the proposed method achieves both the highest classification accuracy and the most compact network structure. The results of the Iris dataset are shown in Figure 2(c).

For the Bupa Liver dataset and WDBC dataset, the proposed method outperforms others in all cases (classification accuracy, number of hidden neurons, standard deviation), as shown in Figure 2(d) and Figure2(h).

For the Australian dataset and Breast Cancer dataset, the DAEM-ELM still maintains the highest classification accuracy and the smallest standard deviation with a medium number of hidden neurons. These results are also confirmed in Figure 2(e) and Figure 2(f).

For the Wine dataset, ABC-ELM is the algorithm with the best performance. The proposed algorithm has the same number of hidden neurons as ABC-ELM, and only slightly poorer performance in accuracy and standard deviation. Figure 2(g) illustrates the results on the Wine dataset.

In summary, the DAEM-ELM algorithm can achieve better performance by using DAEM to select the input weights and hidden biases of the SLFN than the ELM, PSO-ELM, IPSO-ELM, E-ELM and LM algorithms, indicating that the optimal network structure tuned by the DAEM algorithm contributes greatly to the reduction of hidden neurons in the models and a reasonable generalization performance for these datasets.

5. Conclusion

In this paper, a novel extreme learning machine based on electromagnetism-like mechanism (DAEM-ELM) is proposed. In the new algorithm, an improved EM is used to optimize the input weights and hidden biases, and minimum norm least-square scheme is employed to analytically determine the output weights. In the optimization process, the improved EM considers not only the misclassification but also the norm of the output weights as well as constrains the input weights and hidden biases within a reasonable range. In addition, the 5-fold CV method is adopted to prevent the overfitting problem. Experimental results show that DAEM-ELM outperforms other methods (original ELM, PSO-ELM, IPSO-ELM, E-ELM and LM) and has a more compact network structure. It is also confirmed that due to the selection of optimal parameters, the results are more stable with fewer hidden neurons. Hence, it can be concluded the developed DAEM-ELM algorithm can be a feasible and effective algorithm for classification problems. Future research work will be focused on the identification of the optimal hidden neuron number, input weights and hidden biases at the same time.

Acknowledgments

This research work is supported in part by National Natural Science Foundation of China (NSFC) under Grant No. 61603145.

Conflict of interest

All authors declare no conflict of interest in this paper.

References

1. W. Cao, X. Wang, Z. Ming, et al., A review on neural networks with random weights, *Neurocomputing*, (2017), S0925231217314613.
2. G. Camps-Valls, D. Tuia, L. Bruzzone, et al., Advances in hyperspectral image classification: earth monitoring with statistical learning methods, *IEEE Signal Proc. Mag.*, **31** (2013), 45–54.
3. L. Wang, Y. Zeng and T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, *Expert Syst. Appl.*, **42** (2015), 855–863.

4. E. Maggiori, Y. Tarabalka, G. Charpiat, et al., Convolutional neural networks for large-scale remote sensing image classification, *IEEE T. Geosci. Remote,* **55** (2016), 645–657.
5. G. B. Huang, Q. Y. Zhu and C. K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, **70** (2006), 489–501.
6. J. Zhang, Y. F. Lu, B. Q. Zhang, et al., Device-free localization using empirical wavelet transform-based extreme learning machine, Proceedings of the 30th Chinese Control and Decision Conference, (2018), 2585–2590.
7. Y. J. Li, S. Zhang, Y. X. Yin, et al., A soft sensing scheme of gas utilization prediction for blast furnace via improved extreme learning machine, *Neural Process. Lett.* (2018), 10.1007/s11063-018-9888-3.
8. J. Zhang, Y. F. Xu, J. Q. Xue, et al., Real-time prediction of solar radiation based on online sequential extreme learning machine, Proceedings of the 13th IEEE Conference on Industrial Electronics and Applications, (2018), 53–57.
9. R. Z. Song, W. D. Xiao, Q. L. Wei, et al., Neural-network-based approach to finite-time optimal control for a class of unknown nonlinear systems, *Soft Comput.*, **18** (2014), 1645–1653.
10. J. Zhang, W. D. Xiao, Y. J. Li, et al., Multilayer probability extreme learning machine for device-free localization. *Neurocomputing*, (2019), 10.1016/j.neucom.2018.11.106.
11. Y. Park, and H. S. Yang, Convolutional neural network based on an extreme learning machine for image classification, *Neurocomputing*, **339** (2019), 66–76.
12. G. B. Huang, H. Zhou, X. Ding, et al., Extreme learning machine for regression and multiclass classification, *IEEE T. Syst. Man Cy. B.*, **42** (2012), 513–529.
13. F. Han, H. F. Yao and Q. H. Ling, An improved evolutionary extreme learning machine based on particle swarm optimization, *Neurocomputing*, **116** (2013), 87–93.
14. A. Rashno, B. Nazari, S. Sadri, et al., Effective pixel classification of mars images based on ant colony optimization feature selection and extreme learning machine, *Neurocomputing*, **226** (2017), 66–79.
15. G. Li, P. Niu, Y. Ma, et al., Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency, *Knowl-Based Syst.*, **67** (2014), 278–289.
16. İ. B. Ş, and S. Fang, An electromagnetism-like mechanism for global optimization, *J. Global Optim.*, **25** (2003), 263–282.
17. C. J. Zhang, X. Y. Li, L. Gao, et al., An improved electromagnetism-like mechanism algorithm for constrained optimization, *Expert Syst. Appl.*, **40** (2013), 5621–5634.
18. C. T. Tseng, C. H. Lee, Y. S. P. Chiu, et al., A discrete electromagnetism-like mechanism for parallel machine scheduling under a grade of service provision, *Int. J. Prod. Res.*, **55** (2017), 3149–3163.
19. X. Y. Li, L. Gao, Q. K. Pan, et al., An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop, *IEEE T. Syst. Man Cy. Syst.*, (2018), 10.1109/TSMC.2018.2881686.
20. X. Y. Li, C. Lu, L. Gao, et al., An Effective Multi-Objective Algorithm for Energy Efficient Scheduling in a Real-Life Welding Shop, *IEEE T. Ind. Inform.*, **14** (2018), 5400–5409.
21. X. Y. Li, S. Q. Xiao, C. Y. Wang, et al., Mathematical Modeling and a Discrete Artificial Bee Colony Algorithm for the Welding Shop Scheduling Problem, *Memetic Comp.*, (2019), 10.1007/s12293-019-00283-4.

22. Q. Wu, L. Gao, X. Y. Li, et al., Applying an electromagnetism-like mechanism algorithm on parameter optimisation of a multi-pass milling process, *Int. J. Prod. Res.*, **51** (2013), 1777–1788.
23. K. J. Wang, A. M. Adrian, K. H. Chen, et al., An improved electromagnetism-like mechanism algorithm and its application to the prediction of diabetes mellitus, *J. Biomed. Inform.*, **54** (2015), 220–229.
24. S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, *Neural Comput. Appl.*, **27** (2016), 1053–1073.
25. G. Huang, G. B. Huang, S. Song, et al., Trends in extreme learning machines: a review, *Neural Networks*, **61** (2015), 32–48.
26. P. L. Bartlett, The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, *IEEE T. Inform. Theory*, **44** (2002), 525–536.
27. Q. Y. Zhu, A. K. Qin, P. N. Suganthan, et al., Evolutionary extreme learning machine, *Pattern Recogn.*, **38** (2005), 1759–1763.
28. D. Dua, and E. K. Taniskidou, UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science, 2017. Available from: http://archive.ics.uci.edu/ml.
29. Y. Wang, A. Wang, Q. Ai, et al., A novel artificial bee colony optimization strategy-based extreme learning machine algorithm, *Prog. Artif. Intell.*, **6** (2016), 1–12.

AIMS Press