Mathematical Biosciences and Engineering

MBE, 20(2): 3768–3792.
DOI: 10.3934/mbe.2023177
Received: 22 September 2022
Revised: 27 November 2022
Accepted: 28 November 2022
Published: 12 December 2022

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

Research article

Functional extreme learning machine for regression and classification

Xianli Liu[1], Yongquan Zhou[1,2,3,*], Weiping Meng[1] and Qifang Luo[1,3]

[1] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[2] Xiangsihu College of Gunagxi University for Nationalities, Nanning, Guangxi 532100, China
[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

* **Correspondence:** Email: yongquanzhou@126.com; Tel: +8613607882594; Fax: +867713265523.

Abstract: Although Extreme Learning Machine (ELM) can learn thousands of times faster than traditional slow gradient algorithms for training neural networks, ELM fitting accuracy is limited. This paper develops Functional Extreme Learning Machine (FELM), which is a novel regression and classifier. It takes functional neurons as the basic computing units and uses functional equation-solving theory to guide the modeling process of functional extreme learning machines. The functional neuron function of FELM is not fixed, and its learning process refers to the process of estimating or adjusting the coefficients. It follows the spirit of extreme learning and solves the generalized inverse of the hidden layer neuron output matrix through the principle of minimum error, without iterating to obtain the optimal hidden layer coefficients. To verify the performance of the proposed FELM, it is compared with ELM, OP-ELM, SVM and LSSVM on several synthetic datasets, XOR problem, benchmark regression and classification datasets. The experimental results show that although the proposed FELM has the same learning speed as ELM, its generalization performance and stability are better than ELM.

Keywords: functional neurons; functional extreme learning machine; parameter learning algorithm; extreme learning machine

1. Introduction

Artificial Neural Networks (ANNs) simulate the process of human brain information processing through a large number of neurons that are interconnected in a certain way and efficient network learning algorithms [1]. Over the past few decades, ANNs have been widely used in various fields of human needs due to their powerful nonlinear mapping capabilities and parallel computing capabilities [2].

So far, many neural network learning algorithms have been proposed and improved. Jian et al. summarized well-known learning algorithms among them [3]. Among them, the backpropagation algorithm is one of the most mature neural network learning algorithms, and it is also a famous representative of all iterative gradient descent algorithms for supervised learning in neural networks.

It was first proposed by Paul Werbos in the 1970s [4] and widely used after it was rediscovered by Rumelhart and McClelland in 1986 [5]. However, the BP algorithm is not perfect, and there are still inevitable defects, such as easy to fall into a local minimum during the training process, and the convergence speed is slow. In response to these shortcomings, researchers have proposed many methods to improve the backpropagation technique [6]. For example, genetic algorithm is used to optimize the connection weights of BP network [7], add white noise to the weighted sum of BP [8], add momentum term, regularization operator and Adaboost integration algorithm [9], and dynamically change the learning rate according to the change of mean square error [10]. However, these problems of the BP algorithm still restrict its development in many application fields, especially since the learning speed cannot meet the actual needs. Recently, Huang et al. proposed extreme learning machine (ELM), which is a simple and efficient learning algorithm for single hidden layer feedforward neural network (SLFN) [11]. The core idea of the ELM algorithm is that the input weights and hidden layer biases of the network are randomly selected and kept unchanged during the training process, and then the output weights are directly obtained by Moore-Penrose generalized inverse operation. The advantages of ELM are: only the number of hidden layer neurons needs to be optimized, with less human intervention; it avoids the process of iterative optimization of parameters in traditional SLFN training algorithm, which greatly saves training time; the resulting solution is the only optimal solution, which guarantees the generalization performance of the network [12]. Therefore, ELM has been widely used in many fields such as disease diagnosis [13–15], traffic sign recognition [16,17], and prediction [18,19].

In recent years, the significant advantages of ELM have attracted the attention of a large number of researchers in academia and industry, and the research on this algorithm and model has achieved fruitful results [20,21]. The learning process of the standard ELM algorithm can be considered to be based on empirical risk minimization, which tends to produce overfitted models. In addition, since ELM does not consider heteroskedasticity in practical applications, its generalization ability and robustness will be greatly affected when there are many outliers in the training samples. To effectively overcome the above problems, regularization methods are applied to ELM [22–24]. Lu et al. proposed a probabilistic RELM method to reduce the influence of noise data and outliers on the model [25]. Yıldırım and Revan Özkale combined the advantages of Liu estimator and Lasso regression method to deal with the shortcomings of traditional ELM instability and poor generalization [26]. Huang et al. [27] introduced the kernel function into ELM and proposed a general framework-KELM that can be used for regression, binary classification and multi-classification problems, which effectively improved the problem of generalization and stability degradation caused by random parameters. However, kernel selection is an important part of KELM, and it may not always be appropriate to choose empirically. Liu and Wang [28] proposed a multiple kernel extreme learning machine (MK-ELM) to solve this problem. However, MK-ELM cannot effectively handle large-scale datasets, because it needs to optimize more parameters, resulting in high computational complexity of the algorithm. To meet the needs of online real-time applications, online sequential extreme learning machine algorithms (OS-ELM) have been proposed [29], and OS-ELM was improved [30–32]. Online sequential class-specific extreme learning machine (OSCSELM) supports online learning techniques of both chunk-by-chunk and one-by-one learning modes, which is used to solve the class imbalance problem of small and large data sets [33]. Lu et al. used the OS-ELM training method of Kalman filter to improve its stability [34]. OS-ELM was extended to solve the problem of increasing classes [35].

Before the training starts, the user needs to specify the number of hidden layer neurons for ELM. However, how to choose the appropriate number of hidden layer neurons for different applications has always been a difficult and hot topic in the field of neural network research. At present, the methods used by ELM to adjust the hidden layer structure mainly include swarm intelligence optimization [36–40],

incremental method [41–45], pruning [46–50] and adaptive [51–53]. With the advent of the era of big data, storing and processing large-scale data has become an urgent need for enterprises, and the ensemble and parallelism of ELM have therefore become a research hotspot [54–57]. Lam and Wunsch learn features through unsupervised feature learning (UFL) algorithm, and then train features through fast radial basis function (RBF) extreme learning machine (ELM), which improves the accuracy and speed of the algorithm [58]. Yao and Ge proposed distributed parallel extreme learning machine (dp-ELM) and hierarchical extreme learning machine [59]. Duan et al. proposed an efficient ELM with three parallel sub-algorithms based on the Spark framework (SELM) for big data classification. [60]. Many researchers have turned their attention to deep ELM and conducted some innovative research works [61,62]. Dai et al. proposed multilayer one-class extreme learning machine (OC-ELM) [63]. Zhang et al. proposed multi-layer extreme learning machine (ML-ELM) [64]. Yahia et al. proposed a new structure based on extreme learning machine auto-encoder with deep learning structure and a composite wavelet activation function for hidden nodes [65].

The inappropriate initial parameters of the hidden layer (input weights, hidden layer biases and the number of nodes) in the original ELM will lead to poor classification results of ELM [21], although the improved algorithms mentioned above for the original ELM improve its generalization performance, they greatly increase the computational complexity. Therefore, we need a network learning algorithm with fast learning speed and higher generalization performance.

In this paper, we propose a new regression and classification model without iterative optimization parameters in the spirit of extreme learning, called functional extreme learning machine (FELM). FELM aims to use functional neurons (FNs) model as the basic units, and use functional equation-solving theory to guide the modeling process of functional extreme learning machine [66–69]. Like ELM, the FELM parameter matrix is obtained by solving the generalized inverse of the hidden layer neuron output matrix. FELM is a generalization of ELM. Its unique network structure and simple and efficient learning algorithm make it not only solve the problems that ELM can solve, but also solve many problems that ELM cannot solve. However, FELM is also different from ELM. The activation function of ELM is fixed, and ELM has weights and biases. The neuron function of FELM is not fixed, and there are no weights and biases, only parameters (coefficients), so it avoids the influence of random parameters (input weights, hidden layer biases) on the generalization performance and stability of ELM model. Its neuron functions are linear combinations of given basic functions, and the basic functions are selected according to the problem to be solved without specifying the number. The learning essence of FELM is the learning of parameters, and the parameter learning algorithm proposed in this paper does not need iterative calculation and has high accuracy. FELM is compared with other popular technologies in terms of generalization performance and training time on several artificial datasets, benchmark regression and classification datasets. The results show that FELM is not only fast, but also has good generalization performance.

The rest of this paper is organized as follows: Section 2 provides an overview of FN and ELM. In Section 3, the topology of functional extreme learning machine, the theory of structural simplification, and the parameter learning algorithm are described. Section 4 presents the performance comparison results of FELM, classical ELM, OP-ELM, classical SVM and LSSVM on regression and classification problems. Section 5 draws conclusions and discusses future research directions.

2. Preliminaries

Functional neuron model and extreme learning machine (ELM) will be briefly discussed in the following section.

2.1. Functional neuron model

Functional neuron was proposed by Enrique Castillo [66]. Figure 1(a) is a functional neuron model, Figure 1(b) is its expansion model and Figure 1(c) is the expanded model for the green part of Figure 1(b). The mathematical expression of the functional neuron is:

$$O = f(X) \tag{1}$$

where, $X = \{x_1, x_2, \ldots, x_m\}$, $O = \{o_1, o_2, \ldots, o_m\}$, $f(\cdot)$ is functional neuron function, X and O are the input and output of functional neuron function, respectively. Functional neuron function can be expressed by a linear combination of basic functions:

$$f(X) = \sum_{j=1}^{n} a_{ij}\varphi_{ij}(X) = a_i^T \varphi_i(X) \tag{2}$$

where, $\{\varphi_i(X) = (\varphi_{i1}(X), \varphi_{i2}(X), \ldots, \varphi_{in}(X)) | i = 1,2,\ldots,m\}$ is any given basic function family, and different function families can be selected according to specific problems and data. $\varphi_1(X), \ldots, \varphi_m(X)$ are pairwise independent basic function families. Commonly used basic functions are the trigonometric function family and the Fourier family. $\{a_i = (a_{i1}, a_{in}, \ldots, a_{in})^T | i = 1,2,\ldots,m\}$ is a learnable set of parameters.

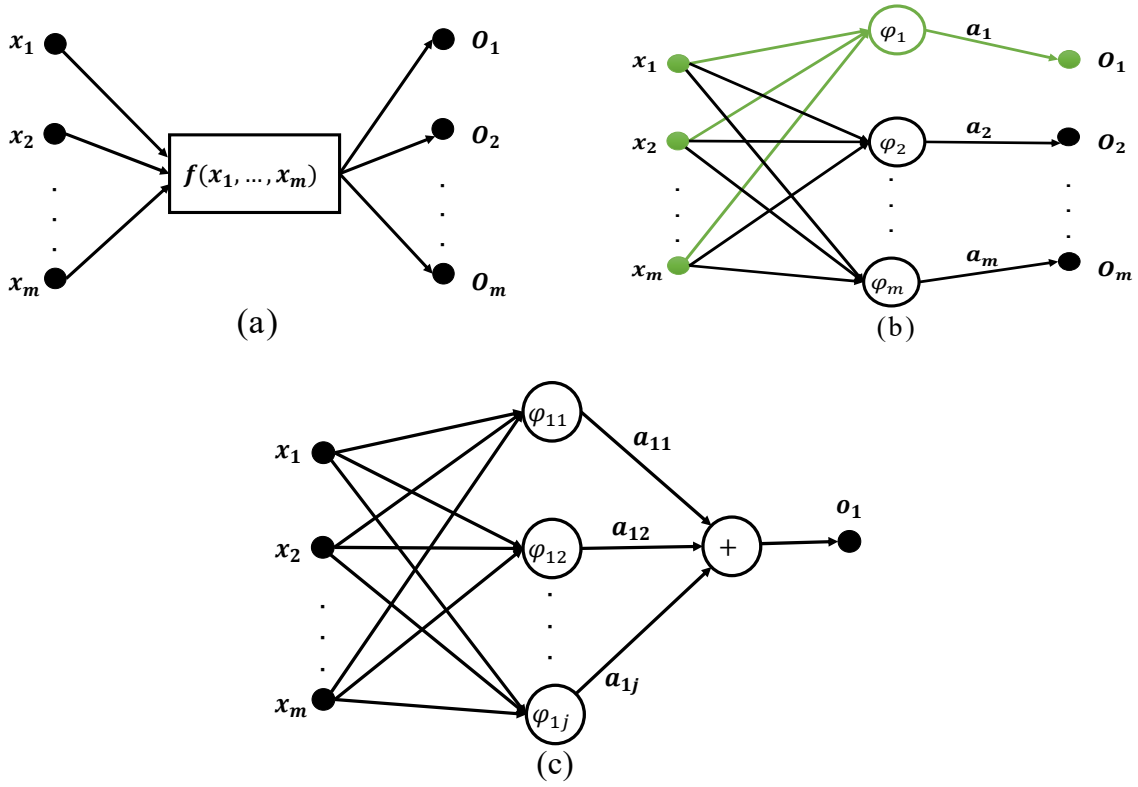


Figure 1. (a) Functional neuron model. (b) Expansion model of functional neuron. (c) The expansion of the green part in (b).

2.2. Extreme learning machine (ELM)

Based on the generalized inverse matrix theory, Huang et al. proposed a new type of single hidden layer feedforward neural network algorithm with excellent performance-extreme learning machine (ELM) [11]. The extreme learning machine network structure is shown in Figure 2.

Given N different training samples $\{x_i, t_i \mid x_i \in R^D, t_i \in R^m, i = 1,2,\ldots,N\}$, $x_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]^T$ as the input vector, $t_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T$ as the corresponding expected output. $g(x)$ is an activation function, which is a nonlinear piecewise continuous function that satisfies the ELM general approximation ability theorem. The commonly used functions are Sigmoid function, Gaussian function, etc. So the mathematical model in Figure 2 is expressed as follows:

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

where, $H = \begin{bmatrix} h_1(x_1) & \cdots & h_L(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_L(x_N) \end{bmatrix}_{N\times L} = \begin{bmatrix} g(\omega_1 \cdot x_1 + b_1) & \cdots & g(\omega_L \cdot x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(\omega_1 \cdot x_N + b_1) & \cdots & g(\omega_L \cdot x_N + b_L) \end{bmatrix}_{N\times L}$.

In ELM, H is called a random feature mapping matrix, $\omega_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{iD}]$ represents the input weight that connects the i^{th} hidden layer neuron and the input layer neuron, b_i represents the bias of the i^{th} hidden layer neuron, and $\beta = [\beta_1, \ldots, \beta_L]^T$ represents the weight matrix between the output layer and the hidden layer. Hidden layer node parameters (ω_i, b_i) are randomly generated and remain unchanged.

Calculate the output weight:

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

where, H^+ represents the Moore-Penrose generalized inverse of the hidden layer output matrix H.

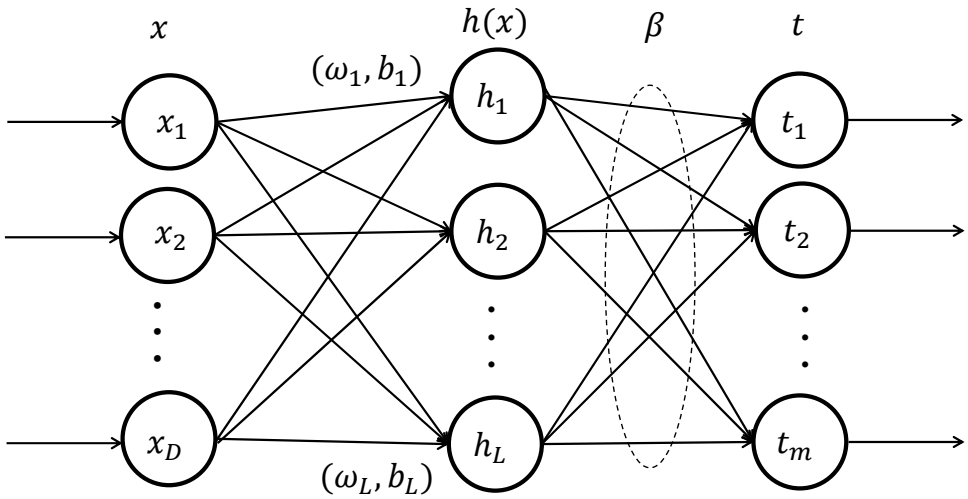


Figure 2. Extreme learning machine network model.

3. Functional extreme learning machine (FELM)

3.1. Functional extreme learning machine topology

According to the example of functional extreme learning machine in Figure 3(a), it can be seen that a functional extreme learning machine network consists of the following elements:

1) Several layers of storage units: One layer of input units ($\{x_1, x_2, x_3\}$), one layer of output storing units ($\{d\}$). Several intermediate storage units ($\{G(x_1, x_2), x_3\}$), they are used to store intermediate information produced by functional neurons. Storage units are represented by solid circles with corresponding names.

2) One or more layers of processing units (i.e., functional neurons): Each functional neuron is a computing unit, which processes the input values from input units or the previous layer of functional neurons, and provides input data to the next layer of neurons or output units. As in Figure 3(a) $\{G, I, F\}$.

3) A set of directed links: They connect the input units to the first layer of processing units, one layer of processing units to the next layer of processing units, and the last layer of computing units to the output units. The arrows indicate the direction in which information flows. Information flows only from the input layer to the output layer.

All these elements together constitute the network architecture of the functional extreme learning machine (FELM). The network architecture corresponds to the functional equation one by one. The functional equation is the key to the FELM learning process. Therefore, the network structure is determined, and the generalization ability of the FELM is also defined.

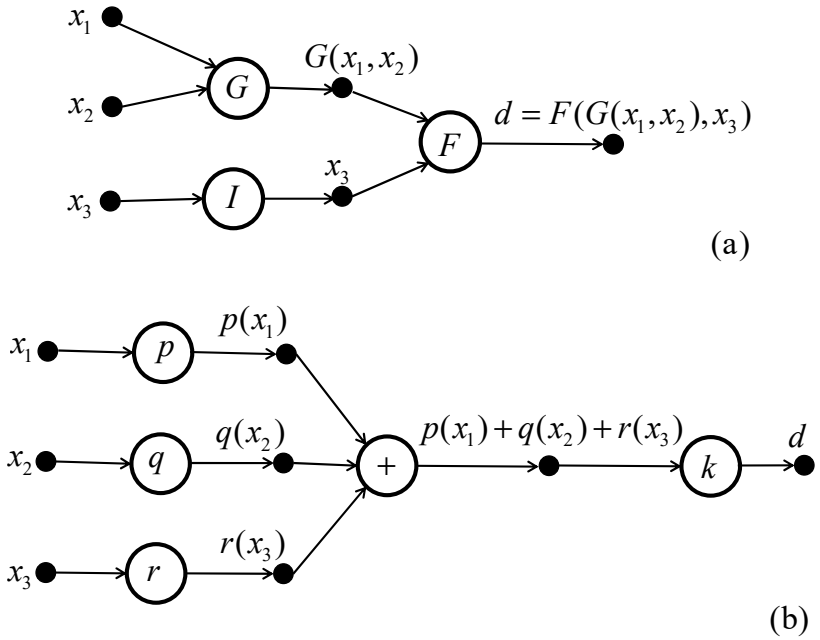


Figure 3. A functional extreme learning machine network structure. (a) The initial structure of the functional extreme learning machine. (b) The equivalent simplified network.

Note the following differences between standard neural networks and FELM networks:

1) The functional neuron as shown in Figure 1(a) is the basic computing unit of FELM. It is different from the M-P neuron (Figure 4), which has no weights $\{w_k\}$ and biases, only parameters, and can have multiple outputs $\{O_1, O_2, \ldots, O_3\}$.

2) In standard neural networks, the functions are given and the weights must be learned. In FELM networks, the functional neuron functions can be linear combinations of any nonlinear correlation basic function families $f(x_1, x_2, \ldots, x_k) = \sum_{j=1}^{n} a_j \varphi_j(x_1, x_2, \ldots, x_k)$. Where, $\{\varphi_j(x_1, x_2, \ldots, x_k) | j = 1,2, \ldots, n\}$ is a given appropriate basic function family, which means that FELM can choose different basic function families for functional neurons depending on the specific problem and data. n represents the number of basic functions. $\{a_j | j = 1,2, \ldots, n\}$ is the learnable parameter set. The following are some commonly used function families: Polynomial family $\{1, x, x^2, \ldots, x^m\}$, Fourier family $\{1, sin(x), cos(x), \ldots, , sin(mx), cos(mx)\}$ and exponential family $\{1, e^x, e^{-x}, \ldots, e^{mx}, e^{-mx}\}$.

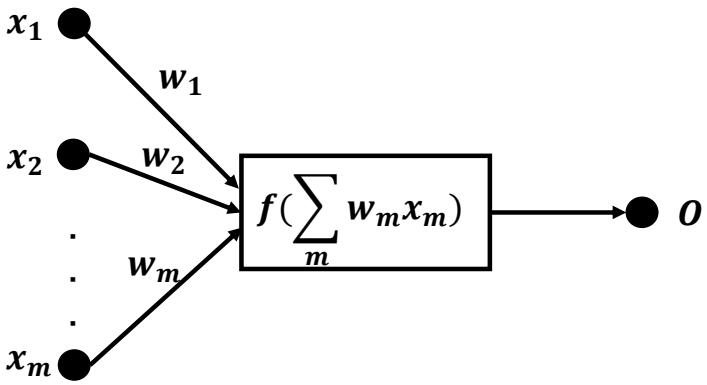


Figure 4. M-P neuron model.

3.2. Structural simplification and unique expression of FELM

Structural simplification: Each initial network structure corresponds to a functional equation set, then the functional equation set solution method is used to simplify the initial structure to obtain an equivalent FELM which is optimal. The functional equation corresponding to Figure 3(a) is:

$$d = F(G(x_1, x_2), x_3) \tag{5}$$

Theorem 1 of [66]: The general solution is continuous on a real rectangle of the functional equation $F[G(x,y),z] = K[x,N(y,z)]$ and is G invertible in two variables. For a fixed value of the second variable, F is invertible in the first variable. For a fixed value of the first variable, K and N are invertible in the second variable:

$$\begin{aligned} F(x,y) &= k[f(x)+g(y)], \quad G(x,y) = f^{-1}[p(x)+q(y)], \\ K(x,y) &= k[p(x)+n(y)], \quad N(x,y) = n^{-1}[q(x)+g(y)], \end{aligned} \tag{6}$$

where, f, k, n, p, q and g are arbitrary continuous and strictly monotonic functions. Therefore, according to Theorem 1 of [66], the general solution of functional Eq (5) is Eq (7):

$$F(x_1, x_2) = k[f(x_1) + r(x_2)], \quad G(x_1, x_2) = f^{-1}[p(x_1) + q(x_2)] \tag{7}$$

According to Eq (7), Eq (5) can be written as:

$$d = F(G(x_1, x_2), x_3) = k[p(x_1) + q(x_2) + r(x_3)] \tag{8}$$

According to Eq (8), the corresponding topology structure can be drawn, as shown in Figure 3(b). Figure 3(b),(a) are equivalent, indicating that they get the same output when they have the same input. In the initial structure of FELM, the functional neuron function is multi-parameter. In the simplified FELM structure, the functional neuron function is a single parameter.

Expression uniqueness of FELM: After structural simplification, the functional equation corresponding to the simplified functional network is $d = k[p(x_1) + q(x_2) + r(x_3)]$, but whether the expression of the functional equation is unique needs to be verified. The following is the verification process, assuming that there are two functional neuron function sets $\{k_1, p_1, q_1, r_1\}$ and $\{k_2, p_2, q_2, r_2\}$, such that :

$$k_1[p_1(x_1) + q_1(x_2) + r_1(x_3)] = k_2[p_2(x_1) + q_2(x_2) + r_2(x_3)]. \forall x_1, x_2, x_3 \tag{9}$$

Let $k_2(v) = k_1(\frac{v-b-c-d}{a})$, then $v = p_2(x_1) + q_2(x_2) + r_2(x_3)$, $\frac{v-b-c-d}{a} = p_1(x_1) + q_1(x_2) + r_1(x_3)$, and $v = ap_1(x_1) + aq_1(x_2) + ar_1(x_3) + b + c + d$. So the solution of the

functional equation is:

$$p_2(x_1) = ap_1(x_1) + b;\ q_2(x_2) = aq_1(x_2) + c;\ r_2(x_3) = ar_1(x_3) + d \tag{10}$$

where a, b, c, d are arbitrary constants. Because any values (a, b, c, d), Eq (10) into Eq (7), will get the following result:

$$F(x_1, x_2) = k_2[ap_1(x_1) + aq_1(x_2) + ar_1(x_3) + b + c + d] = k_1[p_1(x_1) + q_1(x_2) + r_1(x_3)] \tag{11}$$

Therefore, the expression of Eq (8) is unique.

3.3. Functional extreme learning machine learning algorithm

The FELM in Figure 3(b) is taken as an example to illustrate its parameter learning process. Write Eq (8) as follows

$$k^{-1}(x_4) = p(x_1) + q(x_2) + r(x_3), \tag{12}$$

where x_4 represents d.

Each neuron function is a linear combination of given nonlinear correlation basic functions, that is

$$\begin{gathered} p(x_1) = \sum_{j=1}^{m_1} a_{1j}\varphi_{1j}(x_1)\,;\, q(x_2) = \sum_{j=1}^{m_2} a_{2j}\varphi_{2j}(x_2)\,, \\ r(x_3) = \sum_{j=1}^{m_3} a_{3j}\varphi_{3j}(x_3)\,;\, k^{-1}(x_4) = \sum_{j=1}^{m_4} a_{4j}\varphi_{4j}(x_4)\,, \end{gathered} \tag{13}$$

where m_1, m_2, m_3 and m_4 are the numbers of basic functions of p, q, r and k respectively, and a_{ij} is the parameter coefficient of FELM.

Let $a_i = \left[a_{i1}, a_{i2}, \ldots, a_{im_i}\right], i = 1,2,3,4. A = [a_1, a_2, a_3, a_4]^T$ is the parameter to be optimized. $ff_{ir} = \left[\varphi_{i1}(x_{ir}), \varphi_{i2}(x_{ir}), \ldots, \varphi_{im_i}(x_{ir})\right], Ff_r = [ff_{1r}, ff_{2r}, ff_{3r}, ff_{4r}];\ r = 1,2, \ldots, n. Ff = [Ff_1; Ff_2; \ldots; Ff_n];\ B = [k^{-1}(x_{41}), k^{-1}(x_{42}), \ldots, k^{-1}(x_{4n})]^T,\ n$ is the number of observed samples. Then Eq (14) is obtained:

$$Ff \bullet A = B \tag{14}$$

The parameters of FELM can be obtained by Eq (15).

$$A = Ff^{+}B \tag{15}$$

where Ff^{+} is the generalized inverse of Ff.

The above example illustrates the process of model learning. The steps of constructing and simplifying the FELM network and then performing parameter learning are as follows:

Step 1: Based on the characteristics of the problem to be solved, the initial network model is established;

Step 2: Write the functional equation corresponding to the initial network model;

Step 3: Using the functional equation solving method to solve the functional equation and obtain the general solution expression;

Step 4: Based on the general solution expression, use its one-to-one correspondence with the FELM to redraw the corresponding FELM network (simplified FELM);

Step 5: The FELM learning algorithm is used to obtain the optimal parameters of the model.

4. Performance evaluation

In this section, on many benchmark practical problems in the field of function approximation and classification, the performance of the proposed FELM learning algorithm is compared with the

commonly used network algorithms (ELM, OP-ELM, SVM, LS-SVM) on two artificial datasets, 20 different datasets (16 for regression, 4 for classification) and XOR classification problem to verify the effectiveness and superiority of FELM. Experimental environment description for FELM and comparison algorithms: 11th Gen Intel (R) Core (TM) i5-11320H @ 3.20 GHz, 16 GB RAM and MATLAB 2019b. ELM source code used in all experiments: http://www.ntu.edu.sg/home/egbhuang/, OP-ELM source code: https://research.cs.aalto.fi//aml/software.shtml, SVM source code: http://www.csie.ntu.edu.tw/cjlin/libsvm/, and the most popular LS-SVM implementation: http://www.esat.kuleuven.ac.be/sista/lssvmlab/. The sigmoidal activation function is used for ELM, the Gaussian kernel function is used for OP-ELM, and the radial basis function is used for SVM and LS-SVM. The basic functions of our proposed algorithm FELM will be set according to the following specific problems to be solved. In Section 3.1 and Section 3.2, FELM adopts the network structure of Figure 3(b).

It is well known that the performance of SVM is sensitive to the combination of (C, γ). Similar to SVM, the generalization performance of LS-SVM is also closely dependent on the combination of (C, γ). Therefore, in order to achieve good generalization performance, it is necessary to select appropriate cost parameter C and kernel parameter γ for SVM and LS-SVM in each dataset. We tried 17 different values of C and γ, that is, for each dataset, we used 17 different C values and 17 different γ values, a total of 289 pairs (C, γ). Each problem is tested 50 times, and the training data set and the test data set are randomly generated from the entire dataset. For each dataset, two-thirds is the training set and the rest is the test set. This section gives the simulation results, including average training and test accuracy, corresponding standard deviation (Dev), and training time. In experiments, all inputs (attributes) and outputs (targets) have been normalized into [-1, 1].

4.1. Benchmark regression problems

4.1.1. Artificial datasets

To test the performance of FELM on regression problems, we first use the objective function 'sinc' function, which is defined as:

$$y = S\text{inc}(x) = \frac{\sin x}{x}, x \in [-4\pi, 4\pi].$$

To effectively reflect the performance of our algorithm, some different forms of zero mean Gaussian noise pollution are added to the training data points. In particular, we have the following training samples $(x_i, y_i), i = 1,2,\ldots,l$.

(Type A) $y_i = \frac{\sin x_i}{x_i} + \xi_i, x_i \sim U[-4\pi, 4\pi], \xi_i \sim N(0,0.1^2)$

(Type B) $y_i = \frac{\sin x_i}{x_i} + \xi_i, x_i \sim U[-4\pi, 4\pi], \xi_i \sim N(0,0.2^2)$

Next, we proceed to compare the performance of the proposed FELM with other algorithms using the following two synthetic datasets.

$$g(x) = \left|\frac{x-1}{4}\right| + \left|sin\left(\pi\left(1 + \frac{x-1}{4}\right)\right)\right| + 1, -10 \le x \le 10.$$

Again, all training data points are shifted by adding different forms of Gaussian noise below.

(Type C) $y_i = g(x_i) + \xi_i, x_i \sim U[-10,10], \xi_i \sim N(0,0.2^2)$

(Type D) $y_i = g(x_i) + \xi_i, x_i \sim U[-10,10], \xi_i \sim N(0,0.4^2)$

where $U[a, b]$ and $N(c, d^2)$ denote the uniform random variable in $[a, b]$ and the Gaussian random variable with mean c and variance d^2, respectively. The training set and the test set have 5000 data respectively, which are evenly and randomly distributed on the interval $[a, b]$. To make the regression

problem 'real', Types A–D added four different forms of Gaussian noise to all training samples, while the test data remained noise-free.

Table 1. Basic functions given by FELM on 4 synthetic datasets.

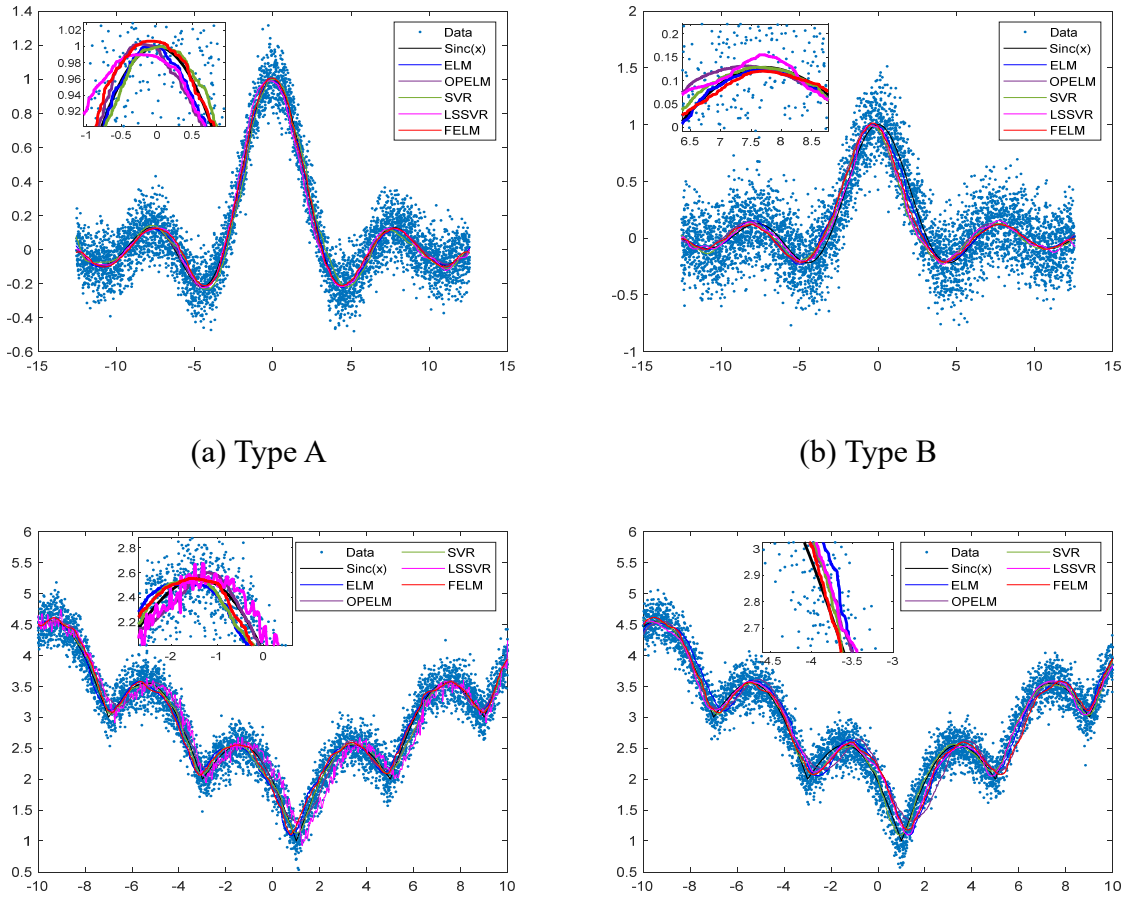
Datasets	Basic functions
Types A and B	$\{1, x^2, x^4, x^6, x^8, x^{10}, cos(x), cos(3x), cos(5x)\}$
Type C	$\{x, x^2, x^3, sin(\frac{x}{2}), cos(\frac{x}{2}), sin(x), cos(x), cos(2x), e^{\frac{x}{3}}, e^{-\frac{x}{3}}, e^{\frac{x}{5}}, cos(\frac{\pi}{2}(1+x)),$ $cos(\pi(1+x)), cos(2\pi(1+x)), cos(3\pi(1+x)), arctan(\frac{x}{7}), arctan(\frac{x}{6}),$ $arctan(\frac{x}{5}), arctan(\frac{x}{4}), arctan(\frac{x}{3}), arctan(\frac{x}{2}), arctan(x), arctan(2x),$ $arctan(3x), arctan(4x), arctan(5x)\}$
Type D	$\{x, sin(\frac{x}{2}), cos(\frac{x}{2}), cos(\frac{x}{3}), sin(x), sin(2x), cos(x), cos(2x), cos(3x), e^{\frac{x}{3}}, e^{-\frac{x}{3}}, e^{\frac{x}{5}},$ $cos(\frac{\pi}{2}(1+x)), cos(\pi(1+x)), cos(2\pi(1+x)), cos(3\pi(1+x))\}$

Table 2. Performance comparison of FELM, ELM, OP-ELM, SVR and LSSVR on four datasets with different types of noises.

Noise	Regressor (C, γ)	Time(s) Training	Testing	Testing RMS	DEV	SVs/nodes
Type A	FELM	0.0024	0.0014	**0.0065**	0.0006	9
	ELM	0.0141	0.0047	0.0065	0.0012	20
	OP-ELM	0.8194	0.0025	0.0060	0.0010	15.50
	SVR (2^7, 2^{-2})	2.2711	0.3150	0.0145	0.0020	1613.46
	LSSVR (2^8, 2^0)	2.0607	0.3824	0.0087	0.0010	5000
Type B	FELM	0.0027	0.0014	**0.0098**	0.0015	9
	ELM	0.0138	0.0084	0.0127	0.0021	20
	OP-ELM	0.7905	0.0024	0.0115	0.0024	14.70
	SVR (2^2, 2^{-2})	1.7248	0.6016	0.0188	0.0026	3089.38
	LSSVR (2^8, 2^0)	1.9938	0.3710	0.0177	0.0018	5000
Type C	FELM	0.0053	0.0024	0.0290	0.0011	26
	ELM	0.4159	0.0194	0.0328	0.0012	90
	OP-ELM	0.8390	0.0031	0.0686	0.0029	19.20
	SVR (2^8, 2^{-1})	35.1289	0.5969	**0.0283**	0.0014	3107.00
	LSSVR (2^8, 2^{-8})	2.2814	0.3572	0.0601	0.0022	5000
Type D	FELM	0.0042	0.0015	**0.0384**	0.0019	16
	ELM	0.4656	0.0209	0.0429	0.0032	100
	OP-ELM	0.7586	0.0027	0.0725	0.0046	18.10
	SVR (2^7, 2^{-1})	15.4804	0.8234	0.0434	0.0040	4020.82
	LSSVR (2^8, 2^0)	1.9812	0.3792	0.0396	0.0037	5000

As shown in Table 1, appropriate basic functions are assigned to our FELM algorithm on four different synthetic datasets. The initial node number of ELM algorithm is 5, and the optimal number of hidden layer nodes is found by interval 5 nodes in 5–100, and the initial maximum number of neurons for OP-ELM is 100. The results of 50 experiments on all algorithms are shown in Table 2, where bold indicates optimal test accuracy. Figure 5 plots the one-time fit curves of FELM and other regressors on these synthetic datasets with different noise types. It can be seen from Table 2 that the

proposed FELM learning algorithm achieves the highest test accuracy (root mean square error, RMS) on artificial datasets with noise types A, B and D. On the artificial dataset with noise type C, FELM is superior to ELM, OP-ELM and LSSVR, and is only lower than SVR. Table 2 also shows the optimal parameter combinations of SVR and LSSVR on these synthetic datasets and the required support vectors (SVs), the network complexity (nodes) of FELM, ELM and OP-ELM. In addition, Table 2 also compares the training and testing time of these five methods. It can be seen that the proposed FELM is the fastest learning method, which is several times or dozens of times faster than ELM, and hundreds of times faster than OP-ELM, SVR and LSSVR. This is because compared with ELM, SVR and LSSVR, FELM has the smallest network complexity, so it requires less learning time. Compared with OP-ELM, FELM does not need to cut out redundant nodes, so it requires less training time. Since the number of support vectors required for SVR and the support vectors required for LSSVR are much larger than the network complexity of FELM, they all take more test time than FELM, at least 60 times more than it does, which means that FELM trained in actual deployment may respond to new external unknown data much faster than SVM and LS-SVM. In short, the proposed FELM outperforms the other four comparison algorithms in approaching four artificial datasets with different types of noise.



(a) Type A

(b) Type B

(c) Type C

(d) Type D

Figure 5. Predictions of different regressors on four synthetic datasets with different forms of noise.

4.1.2. Realistic regression problems

For further evaluation, 16 different regression datasets are selected. These datasets are usually used to test machine learning algorithms, mainly from UCI Machine Learning repository [70] and StatLib [71]. The different attributes of 16 datasets are summarized in Table 3.

As shown in Table 4, we assign appropriate basic functions to our FELM algorithm on 16 different datasets. It can be seen that these basic functions are relatively short in length, indicating that the structural complexity of the networks is low. The initial number of nodes in ELM is 5, and the optimal number of hidden layer nodes is found at intervals of 5 nodes within 5–100. The optimal number of nodes obtained by ELM on each dataset is shown in Table 5. The table also shows the best parameter combination and support vector number of SVR and LSSVR on each dataset, the initial maximum number of neurons and the number of neurons after pruning of OP-ELM.

The results of 50 trials on 16 datasets by the proposed FELM and other comparison algorithms are shown in Tables 6–8. The bold body in Table 6 indicates the optimal test accuracy. The comparison of FELM and the other four comparison algorithms on testing RMSE is shown in Table 6. The minimum test RMSE is obtained on 10 datasets of Autoprice, Balloon, Baskball, Cleveland, Cloud, Diabetes, Machine CPU, Servo, Strike and Wisconsin B.C. On other datasets, although the accuracy obtained by our algorithm is lower than SVR and LSSVR, it is higher than ELM and OP-ELM. The comparison of the five algorithms in training and testing time is shown in Table 7. The table shows that our FELM spends similar training and testing time as ELM, but much less than OP-ELM, SVR and LSSVR. The comparison results of FELM and other algorithms on the standard deviation of testing RMSE are shown in Table 8. According to the table, our FELM is a stable learning method. Figure 6 shows the test RMSE comparison of FELM and other comparison algorithms running 50 times on four datasets (Autoprice, Cleveland, Abalone and Quake). In short, combined with Tables 6–8 and the intuitive display of Figure 6, we can know that the proposed method FELM not only has good versatility and stability, but also has fast training speed.

Table 3. Examples of actual regression.

Datasets	#Train	#Test	#Total	#Features
Abalone	2784	1393	4177	8
Mpg	261	131	392	7
Autoprice	106	53	159	15
Balloon	1334	667	2001	2
Baskball	64	32	96	4
Cleveland	202	101	303	13
Cloud	72	36	108	9
Concrete CS	686	344	1030	8
Diabetes	28	15	43	2
Housing	337	169	506	13
Machine CPU	139	70	209	6
Mg	923	462	1385	6
Quake	1452	726	2178	3
Servo	111	56	167	4
Strike	416	209	625	6
Wisconsin B.C.	129	65	194	32

Table 4. Basic functions given by FELM on 16 regression datasets.

Datasets	Basic functions	Datasets	Basic functions
Abalone	$\{e^{-x}, e^{x}\}$	Diabetes	$\{e^{-x}, e^{x}\}$
Mpg	$\{1, e^{-\frac{x}{2}}, e^{-\frac{x}{3}}, e^{-\frac{x}{4}}\}$	Housing	$\{1, e^{-x}, e^{x}\}$
Autoprice	$\{1, e^{\frac{x}{4}}\}$	Machine CPU	$\{1, e^{-x}, e^{x}\}$
Balloon	$\{e^{-x}, e^{x}, e^{-2x}, e^{2x}, 1, x, x, x^2, x^3\}$	Mg	$\{sin(3x), sin(5x), x^3, e^{\frac{x}{2}}, e^{-\frac{x}{2}}, e^{\frac{x}{5}}, e^{-\frac{x}{5}}\}$
Baskball	$\{e^{-\frac{x}{5}}, e^{-\frac{x}{6}}\}$	Quake	$\{1, sin(x), sin(3x), sin(5x), x, x^3, x^5, e^{-x}, e^{x}\}$
Cleveland	$\{1, e^{-\frac{x}{2}}\}$	Servo	$\{1, e^{-3x}, e^{3x}, sin(\frac{x}{4}), cos(\frac{x}{4}), sin(\frac{x}{6}), cos(\frac{x}{6})\}$
Cloud	$\{1, e^{-\frac{x}{5}}\}$	Strike	$\{1, sin(\frac{x}{4})\}$
Concrete CS	$\{1, sin(x), sin(3x), sin(5x), x, x^2, x^3, e^{-x}, e^{x}\}$	Wisconsin B.C.	$\{sin(\frac{x}{4}), cos(\frac{x}{4})\}$

Table 5. Comparison of network complexity.

Algorithm	SVR			LSSVR		ELM	OP-ELM	
	ε	(C, γ)	# SVs	(C, γ)	# SVs	# nodes	init	final
Abalone	2^{-5}	$(2^{1}, 2^{0})$	1051.28	$(2^{5}, 2^{1})$	2784	35	100	33
Mpg	2^{-8}	$(2^{0}, 2^{-1})$	104.08	$(2^{2}, 2^{1})$	261	30	100	36
Autoprice	2^{-6}	$(2^{3}, 2^{-3})$	42.66	$(2^{6}, 2^{2})$	106	15	100	14
Balloon	2^{-3}	$(2^{7}, 2^{-2})$	6	$(2^{8}, 2^{-1})$	1334	20	100	41
Baskball	2^{-8}	$(2^{3}, 2^{-4})$	44.38	$(2^{2}, 2^{2})$	64	10	62	7
Cleveland	2^{-6}	$(2^{1}, 2^{-7})$	142.72	$(2^{4}, 2^{8})$	198	20	100	9
Cloud	2^{-6}	$(2^{7}, 2^{-8})$	20.62	$(2^{8}, 2^{6})$	72	15	70	20
Concrete CS	2^{-8}	$(2^{4}, 2^{-1})$	174.42	$(2^{7}, 2^{-1})$	686	90	100	87
Diabetes	2^{-5}	$(2^{1}, 2^{-2})$	21.12	$(2^{2}, 2^{0})$	28	5	26	6
Housing	2^{-8}	$(2^{3}, 2^{-2})$	123.42	$(2^{8}, 2^{2})$	337	80	100	58
Machine CPU	2^{-7}	$(2^{8}, 2^{-6})$	8.74	$(2^{7}, 2^{3})$	139	30	100	15
Mg	2^{-6}	$(2^{1}, 2^{0})$	592.84	$(2^{1}, 2^{-2})$	923	70	100	84
Quake	2^{-1}	$(2^{-4}, 2^{7})$	493.28	$(2^{0}, 2^{8})$	1452	30	100	11
Servo	2^{-7}	$(2^{5}, 2^{-3})$	46.36	$(2^{6}, 2^{1})$	111	25	100	41
Strike	2^{-7}	$(2^{8}, 2^{-8})$	88.74	$(2^{-2}, 2^{2})$	416	10	100	11
Wisconsin B.C.	2^{-8}	$(2^{7}, 2^{-8})$	14.92	$(2^{8}, 2^{6})$	129	75	100	51

Table 6. Comparison of testing RMSE.

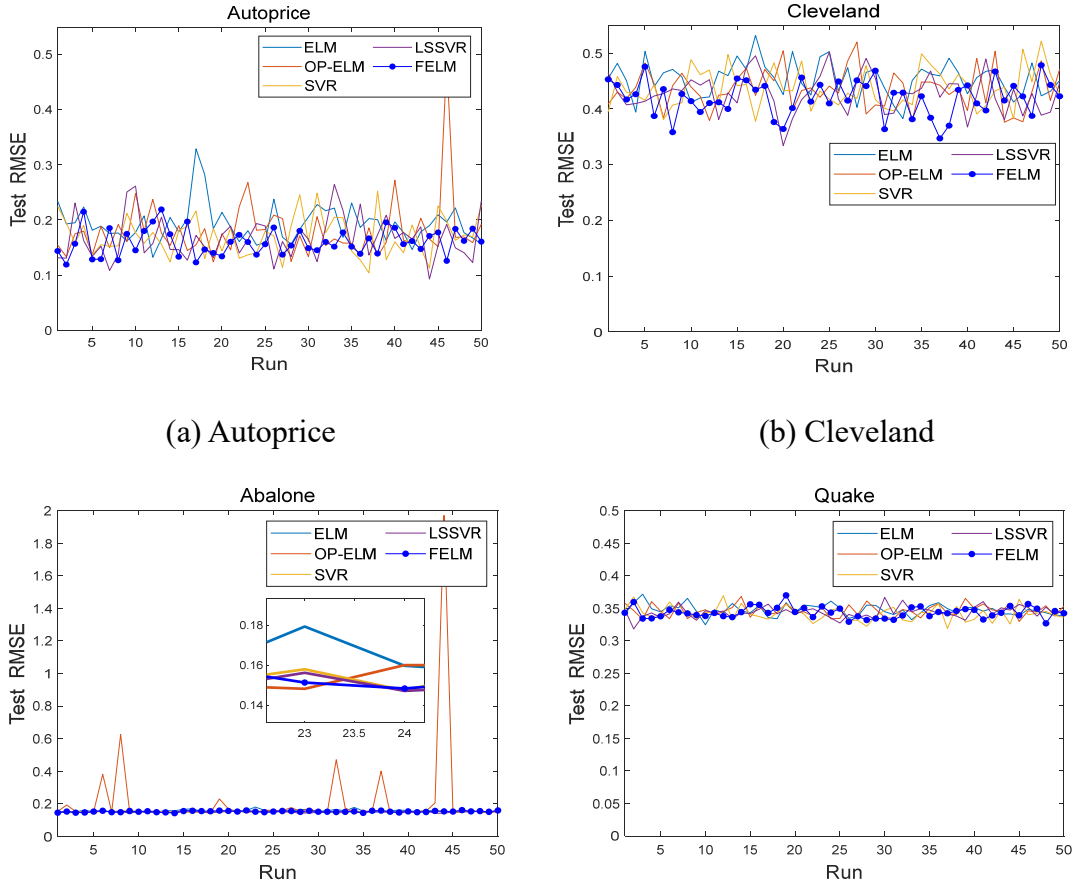
Datasets	ELM	OP-ELM	SVR	LSSVR	FELM
Abalone	0.1566	0.2186	0.1513	**0.1501**	0.1525
Mpg	0.1575	0.1568	0.1403	**0.1389**	0.1415
Autoprice	0.1927	0.1804	0.1712	0.1669	**0.1600**
Balloon	0.0157	0.0175	0.0420	0.0097	**0.0072**
Baskball	0.2631	0.2669	0.2623	0.2576	**0.2477**
Cleveland	0.4538	0.4359	0.4370	0.4281	**0.4204**
Cloud	0.1458	0.1761	0.1240	0.1234	**0.1168**
Concrete CS	0.1249	0.1196	0.1008	**0.0792**	0.1122
Diabetes	0.3787	0.4635	0.3456	0.3242	**0.2938**
Housing	0.1972	0.2305	**0.1458**	0.1478	0.1754
Machine CPU	0.0474	0.0933	0.0731	0.0398	**0.0395**
Mg	0.2748	0.2722	0.2719	**0.2657**	0.2678
Quake	0.3475	0.3466	**0.3420**	0.3441	0.3436
Servo	0.2300	0.1979	0.1819	0.1785	**0.1694**
Strike	0.1517	0.1614	0.1541	0.1423	**0.1320**
Wisconsin B.C.	0.0782	0.0269	0.0670	0.0264	**0.0210**

Table 7. Comparison of training and testing time.

Datasets	ELM (s)		OP-ELM (s)		SVR(s)		LSSVR(s)		FELM (s)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Abalone	0.0020	0.0008	0.5504	0.0019	0.3782	0.0756	0.5977	0.0791	0.0021	0.0006
Mpg	0.0004	0.0004	0.0405	0.0008	0.0040	0.0007	0.0066	0.0017	0.0006	0.0002
Autoprice	0.0002	0.0004	0.0248	0.0003	0.0010	0.0002	0.0037	0.0012	0.0007	0.0004
Balloon	0.0006	0.0005	0.1419	0.0014	0.0015	0.0003	0.1311	0.0203	0.0011	0.0003
Baskball	0.0001	0.0004	0.0108	0.0002	0.0006	0.0001	0.0027	0.0012	0.0001	0.0001
Cleveland	0.0003	0.0007	0.0325	0.0003	0.0047	0.0010	0.0052	0.0018	0.0004	0.0002
Cloud	0.0001	0.0004	0.0136	0.0004	0.0005	0.0001	0.0027	0.0011	0.0002	0.0001
Concrete CS	0.0024	0.0008	0.1046	0.0025	0.0427	0.0028	0.0355	0.0077	0.0033	0.0009
Diabetes	0.0001	0.0004	0.0048	0.0002	0.0002	0.0000	0.0024	0.0011	0.0001	0.0000
Housing	0.0013	0.0005	0.0516	0.0015	0.0099	0.0012	0.0073	0.0029	0.0011	0.0003
Machine CPU	0.0003	0.0004	0.0260	0.0004	0.0004	0.0001	0.0038	0.0020	0.0002	0.0001
Mg	0.0016	0.0006	0.1538	0.0029	0.0628	0.0114	0.0634	0.0089	0.0038	0.0016
Quake	0.0010	0.0005	0.2497	0.0005	0.0656	0.0130	0.1651	0.0186	0.0023	0.0006
Servo	0.0002	0.0004	0.0268	0.0008	0.0027	0.0002	0.0035	0.0011	0.0004	0.0002
Strike	0.0002	0.0003	0.0577	0.0003	0.0139	0.0012	0.0125	0.0039	0.0004	0.0001
Wisconsin B.C.	0.0012	0.0004	0.0289	0.0010	0.0007	0.0001	0.0041	0.0019	0.0014	0.0006

Table 8. Comparison of the standard deviation of testing RMSE.

Datasets	ELM	OP-ELM	SVR	LSSVR	FELM
Abalone	0.0068	0.2685	0.0040	0.0043	0.0043
Mpg	0.0134	0.0176	0.0135	0.0130	0.0123
Autoprice	0.0340	0.0556	0.0364	0.0397	0.0240
Balloon	0.0102	0.0121	0.0121	0.0019	0.0005
Baskball	0.0285	0.0288	0.0293	0.0272	0.0278
Cleveland	0.0312	0.0356	0.0351	0.0343	0.0316
Cloud	0.0309	0.0672	0.0444	0.0363	0.0317
Concrete CS	0.0083	0.0085	0.0085	0.0084	0.0050
Diabetes	0.1095	0.3457	0.0575	0.0524	0.0443
Housing	0.0356	0.1718	0.0192	0.0189	0.0197
Machine CPU	0.0620	0.0623	0.0191	0.0249	0.0221
Mg	0.0100	0.0088	0.0105	0.0079	0.0083
Quake	0.0088	0.0093	0.0112	0.0090	0.0085
Servo	0.0407	0.0442	0.0584	0.0405	0.0423
Strike	0.0383	0.0312	0.0305	0.0380	0.0331
Wisconsin B.C.	0.0183	0.0091	0.0097	0.0030	0.0041



(a) Autoprice (b) Cleveland

(c) Abalone (d) Quake

Figure 6. Comparison of test RMSE of FELM, ELM, OP-ELM, SVR and LSSVR running 50 times on four datasets.

4.2. Benchmark classification problems

4.2.1. XOR problem

The XOR problem dataset randomly generates 1000 samples here, with one class containing 478 samples and the other containing 522 samples. The binary problem is not linearly separable. On this problem, if FELM adopts the structure shown in Figure 3(b), it is not easy to find suitable basic function families for its neurons. Therefore, we adopt the multi-input single-output FELM structure shown in Figure 7 to achieve better generalization performance by increasing the number of hidden layer nodes. FELM sets the number of hidden layer nodes on this problem to be 73, and the node functions correspond to the following:

$\{ sin(x_2)$; $sin(3x_2)$; $sin(5x_2)$; x_2 ; e^{2x_2} ; $sin(x_1)\,sin(x_2)$; $sin(x_1)\,sin(3x_2)$; $sin(x_1)\,sin(5x_2)$; $sin(x_1)x_2$; $sin(x_1)x_2^3$; $sin(x_1)e^{x_2}$; $sin(x_1)e^{x_2}$; $sin(3x_1)\,sin(5x_2)$; $sin(3x_1)x_2$; $sin(3x_1)x_2^2$; $sin(3x_1)e^{-2x_2}$; $sin(3x_1)e^{2x_2}$; $sin(5x_1)\,sin(x_2)$; $sin(5x_1)\,sin(5x_2)$; $sin(5x_1)x_2$; $sin(5x_1)x_2^2$; $sin(5x_1)e^{-x_2}$; $sin(5x_1)e^{2x_2}$; $sin(5x_1)e^{2x_2}$; x_1; $x_1\,sin(x_2)$; $x_1 sin(3x_2)$; $x_1\,sin(5x_2)$; x_1x_2; $x_1x_2^2$; $x_1e^{-2x_2}$; x_1^2; $x_1^2\,sin(3x_2)$; $x_1^2x_2$; $x_1^2x_2^2$; $x_1^2e^{-x_2}$; $x_1^2e^{2x_2}$; $x_1^2e^{2x_2}$; $x_1^3\,sin(x_2)$; $x_1^3\,sin(5x_2)$; $x_1^3x_2$; $x_1^3x_2^2$; $x_1^3e^{-x_2}$; $x_1^3e^{x_2}$; $x_1^3e^{2x_2}$; $e^{-x_1}\,sin(x_2)$; $e^{-x_1}\,sin(3x_2)$; $e^{-x_1}\,sin(5x_2)$; $e^{-x_1}x_2$; $e^{-x_1}x_2^3$; $e^{-x_1}e^{-x_2}$; $e^{-x_1}e^{x_2}$; $e^{-x_1}e^{-2x_2}$; $e^{-x_1}e^{2x_2}$; $e^{x_1}x_2$; $e^{x_1}e^{x_2}$; $e^{x_1}e^{-2x_2}$; e^{-2x_1}; $e^{-2x_1}\,sin(x_2)$; $e^{-2x_1}\,sin(3x_2)$; $e^{-2x_1}\,sin(5x_2)$; $e^{-2x_1}x_2$; $e^{-2x_1}x_2^2$; $e^{-2x_1}e^{-x_2}$; $e^{-2x_1}e^{-2x_2}$; $e^{-2x_1}e^{2x_2}$; $e^{2x_1}x_2$; $e^{2x_1}x_2^2$; $e^{2x_1}x_2^3$; $e^{2x_1}e^{-x_2}$; $e^{2x_1}e^{x_2}$; $e^{2x_1}e^{-2x_2}$; $e^{2x_1}e^{2x_2}\}$.

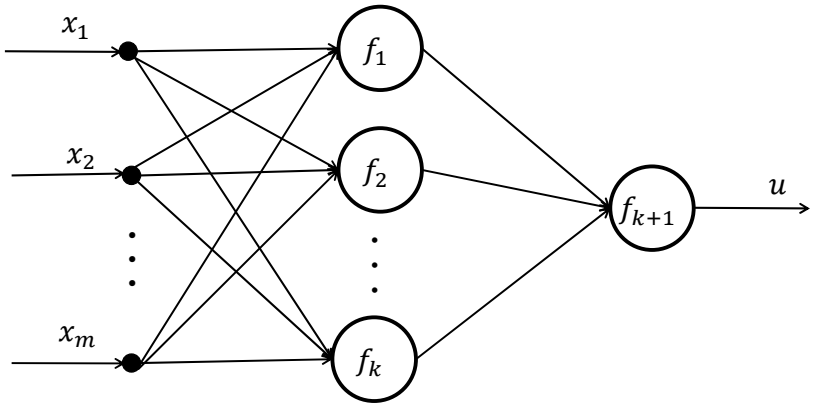
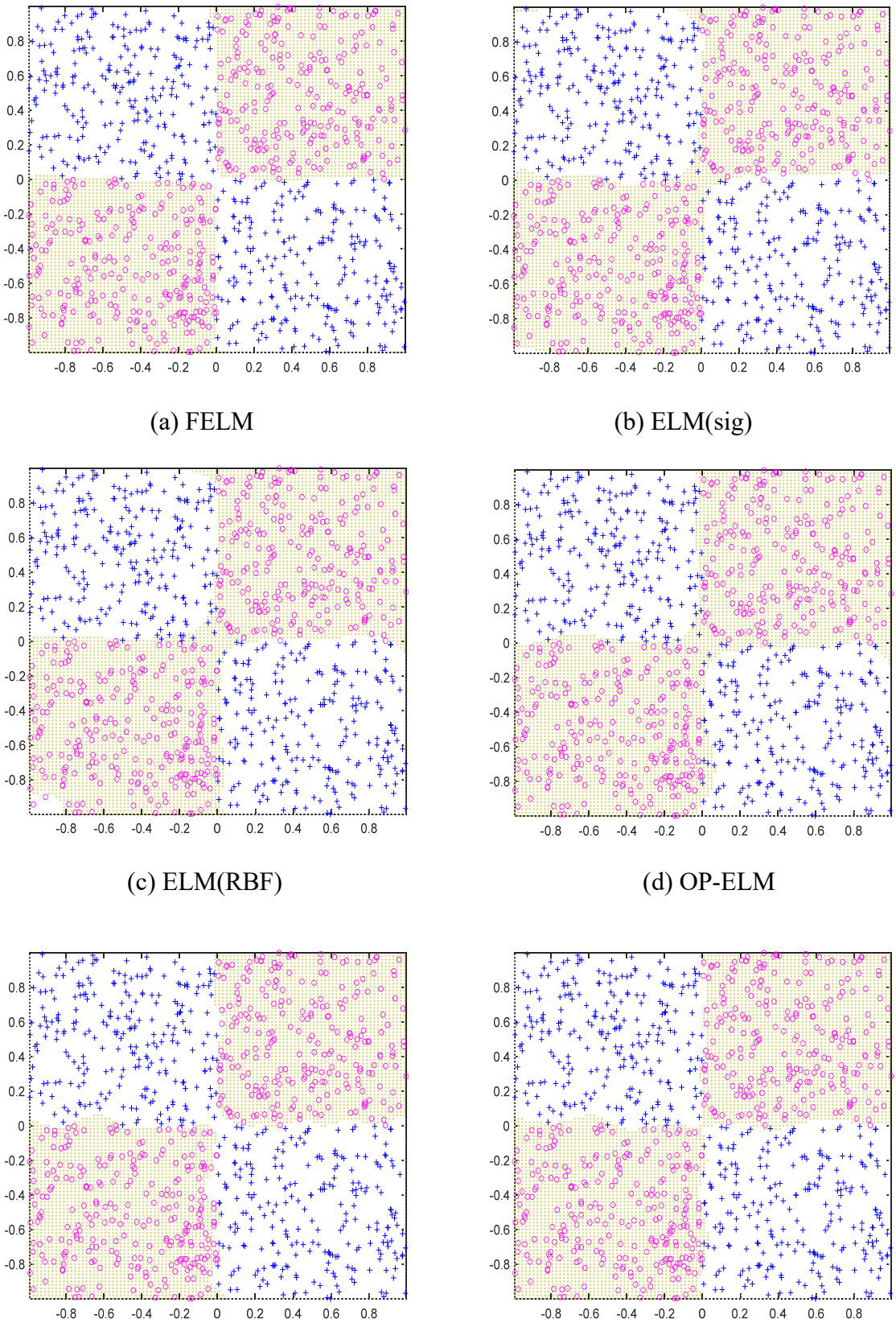


Figure 7. Multiple-input single-output functional extreme learning machine.

Table 9. Performance comparison of FELM, ELM, OP-ELM, SVM and LSSVM on 'XOR' dataset.

Regressor (C, γ)	Time(s)		Testing		SVs/nodes
	Training	Testing	Rate (%)	DEV (%)	
FELM	0.0704	0.0017	98.82	0.62	73
ELM (sig)	0.0054	0.0013	97.29	0.86	155
ELM (RBF)	0.0020	0.0013	97.38	0.90	75
OP-ELM	0.0704	0.0015	97.24	1.06	49
SVM (2^4, 2^6)	0.0188	0.0033	97.29	0.87	269.86
LSSVM (2^3, 2^{-4})	0.0219	0.0062	97.89	0.78	666



(a) FELM (b) ELM(sig)

(c) ELM(RBF) (d) OP-ELM

(e) SVM (f) LSSVM

Figure 8. Separating boundaries of different classifiers in XOR case.

The initial maximum number of neurons for OP-ELM is 100. This section also adds an ELM comparison with an activation function of RBF. The initial number of nodes of the ELM is 5, and the optimal number of hidden layer nodes is found at intervals of 10 nodes within 5–1000, and the optimal number of nodes obtained on each dataset is shown in Table 9. The table also shows the optimal parameter combination and support vector number of SVM and LSSVM on the problem, and the final number of neurons of OP-ELM.

The average results of 50 trials conducted by FELM and other models on the XOR dataset are shown in Table 9. The data in the table show that the performance of FELM is better than ELM, OP-ELM, SVM and LSSVM. Figure 8 shows the boundaries of different classifiers on the XOR problem. It can be seen that, similar to ELM, OP-ELM, SVM and LS-SVM, FELM can solve the XOR problem well.

4.2.2. Realistic classification problems

Table 10. Examples of actual classification. WDBC stands for Wisconsin Breast Cancer. Diabetes stands for Pima Indians Diabetes.

Datasets	#Train	#Test	#Total	#(Featdures/Classes)
Iris	100	50	150	4/3
WDBC	379	190	569	30/2
Diabetes	512	256	768	8/2
Wine	118	60	178	13/3

Table 11. Comparison of network complexity.

Algorithm	FELM	SVM		LSSVM		ELM	OP-ELM	
	basic functions	(C,γ)	# SVs	(C,γ)	# SVs	# nodes	init	final
Iris	$\{1, e^{-x}, e^{x}, e^{-2x}, e^{2x}\}$	$(2^{6},2^{-5})$	20.48	$(2^{-4},2^{3})$	100	15	90	16.80
WDBC	$\{1, e^{-3x}, e^{-5x}\}$	$(2^{2},2^{-3})$	55.04	$(2^{2},2^{5})$	379	65	90	32.90
Diabetes	$\{e^{-x}, e^{x}\}$	$(2^{4},2^{-5})$	281.7	$(2^{4},2^{3})$	512	15	90	19.90
Wine	$\{e^{-x}, e^{-2x}\}$	$(2^{-1},2^{-2})$	63.64	$(2^{-3},2^{3})$	118	15	90	33.40

Table 12. Comparison of testing correct classification Rate.

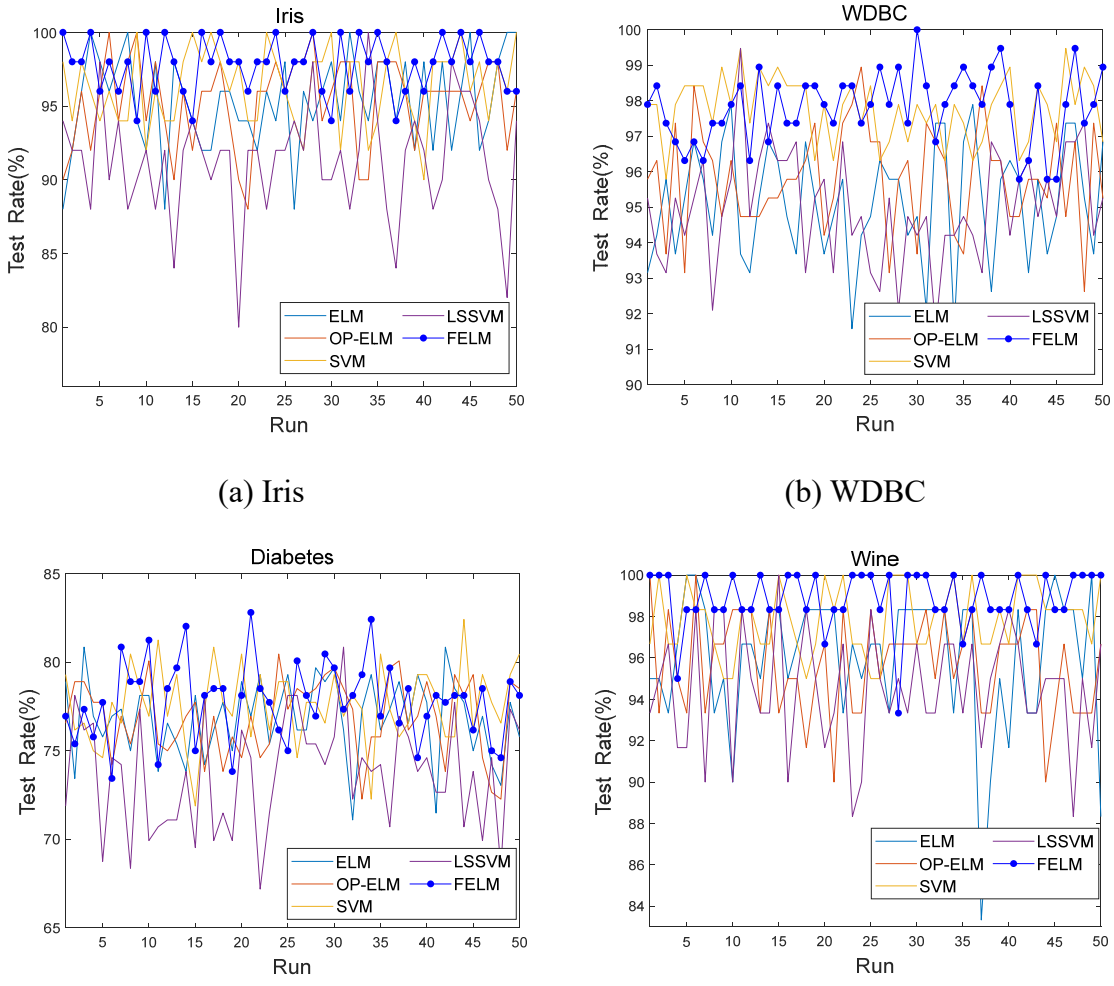
Datasets	ELM	OP-ELM	SVM	LSSVM	FELM
Iris	95.36	95.36	96.40	91.32	**97.76**
WDBC	95.16	95.83	97.78	95.01	**97.81**
Diabetes	76.83	76.89	77.59	73.70	**77.85**
Wine	96.03	95.73	97.90	94.50	**98.87**

Table 13. Comparison of training and testing time.

Datasets	ELM (s)		OP-ELM (s)		SVM (s)		LSSVM (s)		FELM (s)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Iris	0.0002	0.0005	0.0217	0.0004	0.0004	0.0001	0.0022	0.0036	0.0002	0.0001
WDBC	0.0013	0.0009	0.0519	0.0010	0.0047	0.0010	0.0079	0.0161	0.0043	0.0010
Diabetes	0.0004	0.0007	0.0716	0.0006	0.0199	0.0044	0.0120	0.0078	0.0005	0.0002
Wine	0.0002	0.0005	0.0255	0.0007	0.0011	0.0003	0.0024	0.0076	0.0004	0.0002

Table 14. Comparison of the standard deviation of testing correct classification rate.

Datasets	ELM	OP-ELM	SVM	LSSVM	FELM
Iris	3.19	2.95	2.62	3.89	1.84
WDBC	1.63	1.45	0.91	1.56	1.00
Diabetes	2.21	2.15	2.15	3.15	2.17
Wine	3.31	2.43	1.64	2.82	1.41



(a) Iris (b) WDBC

(c) Diabetes (d) Wine

Figure 9. Comparison of test successful classification rates for FELM, ELM, OP-ELM, SVM and LSSVM on four datasets run 50 times.

The newly proposed FELM algorithm is compared with four other popular algorithms (ELM, OP-ELM, SVM and LSSVM) on four classification problems: Iris, WDBC, Diabetes and Wine. These four datasets are from UCI Machine Learning repository [70], and the number of samples, attributes and classes are shown in Table 10. The ELM algorithm sets the initial number of nodes to 5, and finds the optimal number of hidden layer nodes at intervals of 10 nodes within 5–1000. As shown in Table 11, we assign appropriate basic functions to these datasets for our FELM algorithm, and the optimal number of nodes obtained by the ELM algorithm on each dataset. The table also shows the optimal parameter combination and support vector number of SVM and LSSVM, the initial maximum number of neurons and the number of neurons after pruning of OP-ELM.

The performance comparison between all algorithms is shown in Tables 12–14. In the comparison of these five algorithms, obviously, better test results are given in bold. In Table 12, our FELM compared with four other algorithms on testing correct classification rate, and FELM achieved the highest test correct classification rate. The comparison results of the five algorithms for training and testing time are shown in Table 13. The learning speed of FELM is similar to the ELM, which is much faster than the OP-ELM, SVM and LSSVM. In Table 14, FELM is compared with other comparison algorithms on the standard deviation of testing correct classification rate, and the results show that FELM has good stability. Figure 9 shows the successful classification rate comparison of FELM and the other four algorithms running 50 times on four classification datasets. It can be seen that FELM obtains the highest number of higher classification rates. Compared with other algorithms, the curve fluctuation is smaller, indicating that its stability is better. In summary, combined with Tables 12–14 and Figure 9, it can be seen that FELM not only guarantees the learning speed in all cases, but also achieves better generalization performance.

5. Conclusions and future works

In this paper, we propose a new method for data regression and classification called functional extreme learning machine (FELM). Different from the traditional ELM, FELM is problem-driven rather than model-driven, without the concept of weight and bias. It uses the functional neuron as the basic unit, and uses functional equation solving theory to guide its modeling process. The functional neuron of the learning machine is represented by a linear combination of any linearly independent basic functions, and infinitely approximates the desired accuracy by adjusting the coefficients of the basic functions in the functional neuron. In addition, the parameter fast learning algorithm proposed in this paper does not need iteration and has high accuracy. Its learning process is different from the ELM used by people at present. It is expected to fundamentally overcome the shortcomings of the random initial parameters of the hidden layer (connection weights, bias values, number of nodes) in the current ELM theory that significantly affect the classification accuracy of ELM. Like ELM, FELM has less human intervention. It only needs to match the appropriate basic functions for the problem, and can obtain the optimal parameters according to the parameter learning algorithm without iteration. Simulation results show that compared with ELM, FELM has better performance and similar learning speed in regression and classification. Compared to SVM and LS-SVM, FELM can run stably with faster learning speed (up to several hundred times) while guaranteeing generalization performance. The proposed FELM theory provides a new idea for tapping the potential of extreme learning and broadening the application of extreme learning, which has important theoretical significance and broad application prospects. In the future work, we will use the parameter screening algorithm to further improve the generalization ability and stability of FELM and broaden its practical application range. These are the author's next works.

Acknowledgments

This research is funded by the National Natural Science Foundation of China, Grant number 62066005, U21A20464. Project of the Guangxi Science and Technology under Grant No. 2019KY0185.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. L. C. Jiao, S. Y. Yang, F. Liu, S. G. Wang, Z. X. Feng, Seventy years beyond neural networks: retrospect and prospect, *Chin. J. Comput.*, **39** (2016), 1697–1716.
2. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: A survey, *Heliyon*, **4** (2018), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938
3. A. K. Jain, J. Mao, K. M. Mohiuddin, Artificial neural networks: A tutorial, *Computer*, **29** (1996), 31–44. https://doi.org/10.1109/2.485891
4. P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D thesis, Harvard University, Boston, USA, 1974.
5. D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, USA, 1986. https://doi.org/10.7551/mitpress/5236.001.0001
6. K. Vora, S. Yagnik, M. Scholar, A survey on backpropagation algorithms for feedforward neural networks, *Int. J. Eng. Dev. Res.*, **1** (2014), 193–197.
7. S. Ding, C. Su, J. Yu, An optimizing BP neural network algorithm based on genetic algorithm, *Artif. Intell. Rev.*, **36** (2011), 153–162. https://doi.org/10.1007/s10462-011-9208-z
8. A. Sapkal, U. V. Kulkarni, Modified backpropagation with added white Gaussian noise in weighted sum for convergence improvement, *Procedia Comput. Sci.*, **143** (2018), 309–316. https://doi.org/10.1016/j.procs.2018.10.401
9. W. Yang, X. Liu, K. Wang, J. Hu, G. Geng, J. Feng, Sex determination of three-dimensional skull based on improved backpropagation neural network, *Comput. Math. Methods Med.*, **2019** (2019), 1–8. https://doi.org/10.1155/2019/9163547
10. W. C. Pan, S. D. Liu, Optimization research and application of BP neural network, *Comput. Technol. Dev.*, **29** (2019), 74–76. https://doi.org/10.3969/j.issn.1673-629X.2019.05.016
11. G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme learning machine: theory and applications, *Neurocomputing*, **70** (2006), 489–501. https://doi.org/10.1016/j.neucom.2005.12.126
12. S. Y. Lu, Z. H. Lu, S. H. Wang, Y. D. Zhang, Review of extreme learning machine, *Meas. Control Technol.*, **37** (2018), 3–9. https://doi.org/10.19708/j.ckjs.2018.10.001
13. F. Mohanty, S. Rup, B. Dash, B. Majhi, M. N. S. Swamy, A computer-aided diagnosis system using Tchebichef features and improved grey wolf optimized extreme learning machine, *Appl. Intell*., **49** (2019), 983–1001. https://doi.org/10.1007/s10489-018-1294-z
14. D. Muduli, R. Dash, B. Majhi, Automated breast cancer detection in digital mammograms: A moth flame optimization based ELM approach, *Biomed. Signal Process. Control*, **59** (2020), 101912. https://doi.org/10.1016/j.bspc.2020.101912
15. Z. Wang, Y. Luo, J. Xin, H. Zhang, L. Qu, Z. Wang, et al., Computer-aided diagnosis based on extreme learning machine: a review, *IEEE Access*, **8** (2020), 141657–141673. https://doi.org/10.1109/ACCESS.2020.3012093
16. Z. Huang, Y. Yu, J. Gu, H. Liu, An efficient method for traffic sign recognition based on extreme learning machine, *IEEE Trans. Cybern.*, **47** (2016), 920–933. https://doi.org/10.1109/TCYB.2016.2533424
17. S. Aziz, E. A. Mohamed, F. Youssef, Traffic sign recognition based on multi-feature fusion and ELM classifier, *Procedia Comput. Sci.*, **127** (2018), 146–153. https://doi.org/10.1016/j.procs.2018.01.109

18. Z. M. Yaseen, S. O. Sulaiman, R. C. Deo, K. W. Chau, An enhanced extreme learning machine model for river flow forecasting: State-of-the-art, practical applications in water resource engineering area and future research direction, *J. Hydrol.*, **569** (2019), 387–408. https://doi.org/10.1016/j.jhydrol.2018.11.069
19. M. Shariati, M. S. Mafipour, B. Ghahremani, F. Azarhomayun, M. Ahmadi, M. T. Trung, et al., A novel hybrid extreme learning machine-grey wolf optimizer (ELM-GWO) model to predict compressive strength of concrete with partial replacements for cement, *Eng. Comput.*, **38** (2020), 757–779. https://doi.org/10.1007/s00366-020-01081-0
20. R. Xu, X. Liang, J. S. Qi, Z. Y. Li, S. S. Zhang, Advances and trends in extreme learning machine, *Chin. J. Comput.*, **42** (2019), 1640–1670. https://doi.org/10.11897/SP.J.1016.2019.01640
21. J. Wang, S. Lu, S. H. Wang, Y. D. Zhang, A review on extreme learning machine, *Multimedia Tools Appl.*, **81** (2022), 41611–41660. https://doi.org/10.1007/s11042-021-11007-7
22. W. Deng, Q. Zheng, L. Chen, Regularized extreme learning machine, in *2009 IEEE Symposium on Computational Intelligence and Data Mining*, IEEE, Nashville, USA, (2009), 389–395. https://doi.org/10.1109/CIDM.2009.4938676
23. Y. P. Zhao, Q. K. Hu, J. G. Xu, B. Li, G. Huang, Y. T. Pan, A robust extreme learning machine for modeling a small-scale turbojet engine, *Appl. Energy*, **218** (2018), 22–35. https://doi.org/10.1016/j.apenergy.2018.02.175
24. K. Wang, H. Pei, J. Cao, P. Zhong, Robust regularized extreme learning machine for regression with non-convex loss function via DC program, *J. Franklin Inst.*, **357** (2020), 7069–7091. https://doi.org/10.1016/j.jfranklin.2020.05.027
25. X. Lu, L. Ming, W. Liu, H. X. Li, Probabilistic regularized extreme learning machine for robust modeling of noise data, *IEEE Trans. Cybern.*, **48** (2018), 2368–2377. https://doi.org/10.1109/TCYB.2017.2738060
26. H. Yıldırım, M. Revan Özkale, LL-ELM: A regularized extreme learning machine based on L_1-norm and Liu estimator, *Neural Comput. Appl.*, **33** (2021), 10469–10484. https://doi.org/10.1007/s00521-021-05806-0
27. G. B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, **42** (2012), 513–529. https://doi.org/10.1109/TSMCB.2011.2168604
28. X. Liu, L. Wang, G. B. Huang, J. Zhang, J. Yin, Multiple kernel extreme learning machine, *Neurocomputing*, **149** (2015), 253–264. https://doi.org/10.1016/j.neucom.2013.09.072
29. N. Y. Liang, G. B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, *IEEE Trans. Neural Networks*, **17** (2006), 1411–1423. https://doi.org/10.1109/TNN.2006.880583
30. J. Yang, F. Ye, H. J. Rong, B. Chen, Recursive least mean p-power extreme learning machine, *Neural Networks*, **91** (2017), 22–33. https://doi.org/10.1016/j.neunet.2017.04.001
31. J. Yang, Y. Xu, H. J. Rong, S. Du, B. Chen, Sparse recursive least mean p-power extreme learning machine for regression, *IEEE Access*, **6** (2018), 16022–16034. https://doi.org/10.1109/ACCESS.2018.2815503
32. S. Ding, B. Mirza, Z. Lin, J. Cao, X. Lai, T. V. Nguyen, et al., Kernel based online learning for imbalance multiclass classification, *Neurocomputing*, **277** (2018), 139–148. https://doi.org/10.1016/j.neucom.2017.02.102
33. S. Shukla, B. S. Raghuwanshi, Online sequential class-specific extreme learning machine for binary imbalanced learning, *Neural Networks*, **119** (2019), 235–248. https://doi.org/10.1016/j.neunet.2019.08.018

34. F. Lu, J. Wu, J. Huang, X. Qiu, Aircraft engine degradation prognostics based on logistic regression and novel OS-ELM algorithm, *Aerosp. Sci. Technol.*, **84** (2019), 661–671. https://doi.org/10.1016/j.ast.2018.09.044
35. H. Yu, H. Xie, X. Yang, H. Zou, S. Gao, Online sequential extreme learning machine with the increased classes, *Comput. Electr. Eng.*, **90** (2021), 107008. https://doi.org/10.1016/j.compeleceng.2021.107008
36. Q. Y. Zhu, A. K. Qin, P. N. Suganthan, G. B. Huang, Evolutionary extreme learning machine, *Pattern Recognit.*, **38** (2005), 1759–1763. https://doi.org/10.1016/j.patcog.2005.03.028
37. D. T. Bui, P. T. T. Ngo, T. D. Pham, A. Jaafari, N. Q. Minh, P. V. Hoa, et al., A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping, *Catena*, **179** (2019), 184–196. https://doi.org/10.1016/j.catena.2019.04.009
38. W. Cai, J. Yang, Y. Yu, Y. Song, T. Zhou, J. Qin, PSO-ELM: A hybrid learning model for short-term traffic flow forecasting, *IEEE Access*, **8** (2020), 6505–6514. https://doi.org/10.1109/ACCESS.2019.2963784
39. J. Zeng, B. Roy, D. Kumar, A. S. Mohammed, D. J. Armaghani, J. Zhou, et al., Proposing several hybrid PSO-extreme learning machine techniques to predict TBM performance, *Eng. Comput.*, **38** (2022), 3811–3827. https://doi.org/10.1007/s00366-020-01225-2
40. R. M. Adnan, R. R. Mostafa, O. Kisi, Z. M. Yaseen, S. Shahid, M. Zounemat-Kermani, Improving streamflow prediction using a new hybrid ELM model combined with hybrid particle swarm optimization and grey wolf optimization, *Knowl.-Based Syst.*, **230** (2021), 107379. https://doi.org/10.1016/j.knosys.2021.107379
41. G. B. Huang, L. Chen, C. K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, *IEEE Trans. Neural Networks*, **17** (2006), 879–892. https://doi.org/10.1109/tnn.2006.875977
42. G. B. Huang, M. B. Li, L. Chen, C. K. Siew, Incremental extreme learning machine with fully complex hidden nodes, *Neurocomputing*, **71** (2008), 576–583. https://doi.org/10.1016/j.neucom.2007.07.025
43. Y. X. Wu, D. Liu, H. Jiang, Length-changeable incremental extreme learning machine, *J. Comput. Sci. Technol.*, **32** (2017), 630–643. https://doi.org/10.1007/s11390-017-1746-7
44. S. Song, M. Wang, Y. Lin, An improved algorithm for incremental extreme learning machine, *Syst. Sci. Control Eng.*, **8** (2020), 308–317. https://doi.org/10.1080/21642583.2020.1759156
45. H. C. Leung, C. S. Leung, E. W. M. Wong, Fault and noise tolerance in the incremental extreme learning machine, *IEEE Acces*s, **7** (2019), 155171–155183. https://doi.org/10.1109/ACCESS.2019.2948059
46. H. J. Rong, Y. S. Ong, A. H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, *Neurocomputing*, **72** (2008), 359–366. https://doi.org/10.1016/j.neucom.2008.01.005
47. Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, *IEEE Trans. Neural Networks*, **21** (2010), 158–162. https://doi.org/10.1109/TNN.2009.2036259
48. R. M. Adnan, Z. Liang, S. Trajkovic, M. Zounemat-Kermani, B. Li, O. Kisi, Daily streamflow prediction using optimally pruned extreme learning machine, *J. Hydrol.*, **577** (2019), 123981. https://doi.org/10.1016/j.jhydrol.2019.123981
49. Q. Fan, L. Niu, Q. Kang, Q. Kang, Regression and multiclass classification using sparse extreme learning machine via smoothing group $L_{1/2}$ regularizer, *IEEE Access*, **8** (2020), 191482–191494. https://doi.org/10.1109/ACCESS.2020.3031647

50. B. Li, Y. P. Zhao, Group reduced kernel extreme learning machine for fault diagnosis of aircraft engine, *Eng. Appl. Artif. Intell.*, **96** (2020), 103968. https://doi.org/10.1016/j.engappai.2020.103968
51. G. Feng, Y. Lan, X. Zhang, Z. Qian, Dynamic adjustment of hidden node parameters for extreme learning machine, *IEEE Trans. Cybern.*, **45** (2015), 279–288. https://doi.org/10.1109/TCYB.2014.2325594
52. G. Zeng, B. Zhang, F. Yao, S. Chai, Modified bidirectional extreme learning machine with Gram-Schmidt orthogonalization method, *Neurocomputing*, **316** (2018), 405–414. https://doi.org/10.1016/j.neucom.2018.08.029
53. M. Pratama, G. Zhang, M. J. Er, S. Anavatti, An incremental type-2 meta-cognitive extreme learning machine, *IEEE Trans. Cybern.*, **47** (2017), 339–353. https://doi.org/10.1109/TCYB.2016.2514537
54. Z. Chen, C. Jiang, L. Xie, A novel ensemble ELM for human activity recognition using smartphone sensors, *IEEE Trans. Ind. Inf.*, **15** (2019), 2691–2699. https://doi.org/10.1109/TII.2018.2869843
55. S. F. Stefenon, R. B. Grebogi, R. Z. Freire, A. Nied, L. H. Meyer, Optimized ensemble extreme learning machine for classification of electrical insulators conditions, *IEEE Trans. Ind. Electron.*, **67** (2020), 5170–5178. https://doi.org/10.1109/TIE.2019.2926044
56. X. B. Wang, X. Zhang, Z. Li, J. Wu, Ensemble extreme learning machines for compound-fault diagnosis of rotating machinery, *Knowl.-Based Syst.*, **188** (2020), 105012. https://doi.org/10.1016/j.knosys.2019.105012
57. X. Zhou, Y. Zhang, Ensemble extreme learning machine approach to thermal infrared subpixel temperature estimation, *IEEE Geosci. Remote Sens. Lett.*, **18** (2021), 920–924. https://doi.org/10.1109/LGRS.2020.2985500
58. D. Lam, D. Wunsch, Unsupervised feature learning classification with radial basis function extreme learning machine using graphic processors, *IEEE Trans. Cybern.*, **47** (2017), 224–231. https://doi.org/10.1109/TCYB.2015.2511149
59. L. Yao, Z. Ge, Distributed parallel deep learning of hierarchical extreme learning machine for multimode quality prediction with big process data, *Eng. Appl. Artif. Intell.*, **81** (2019), 450–465. https://doi.org/10.1016/j.engappai.2019.03.011
60. M. Duan, K. Li, X. Liao, K. Li, A parallel multiclassification algorithm for big data using an extreme learning machine, *IEEE Trans. Neural Networks Learn. Syst.*, **29** (2018), 2337–2351. https://doi.org/10.1109/TNNLS.2017.2654357
61. J. Tang, C. Deng, G. B. Huang, Extreme learning machine for multilayer perceptron, *IEEE Trans. Neural Networks Learn. Syst.*, **27** (2016), 809–821. https://doi.org/10.1109/TNNLS.2015.2424995
62. C. M. Wong, C. M. Vong, P. K. Wong, J. Cao, Kernel-based multilayer extreme learning machines for representation learning, *IEEE Trans. Neural Networks Learn. Syst.*, **29** (2018), 757–762. https://doi.org/10.1109/TNNLS.2016.2636834
63. H. Dai, J. Cao, T. Wang, M. Deng, Z. Yang, Multilayer one-class extreme learning machine, *Neural Networks*, **115** (2019), 11–22. https://doi.org/10.1016/j.neunet.2019.03.004
64. J. Zhang, Y. Li, W. Xiao, Z. Zhang, Non-iterative and fast deep learning: Multilayer extreme learning machines, *J. Franklin Inst.*, **357** (2020), 8925–8955. https://doi.org/10.1016/j.jfranklin.2020.04.033

65. S. Yahia, S. Said, M. Zaied, Wavelet extreme learning machine and deep learning for data classification, *Neurocomputing*, **470** (2022), 280–289. https://doi.org/10.1016/j.neucom.2020.04.158
66. E. Castillo, Functional networks, *Neural Process. Lett.*, **7** (1998), 151–159. https://doi.org/10.1023/A:1009656525752
67. Y. Q. Zhou, L. C. Jiao, Universal learning algorithm of hierarchical function networks, *Chin. J. Comput.*, **28** (2005), 1277–1286. https://doi.org/10.3321/j.issn:0254-4164.2005.08.004
68. Y. Q. Zhou, B. Zhao, L. C. Jiao, Serial function networks method and learning algorithm with applications, *Chin. J. Comput.*, **31** (2008), 1073–1081.
69. G. Zhou, Y. Zhou, H. Huang, Z. Tang, Functional networks and applications: A survey, *Neurocomputing*, **335** (2019), 384–399. https://doi.org/10.1016/j.neucom.2018.04.085
70. A. Asuncion, D. J. Newman, UCI machine learning repository, School of Information and Computer Science, University of California, Irvine, CA, 2007. Available from: http://www.ics.uci.edu/~mlearn/MLRepository.html.
71. StatLib DataSets Archive. Available from: http://lib.stat.cmu.edu/datasets.

AIMS Press