This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Inverse-Free Extreme Learning Machine With Optimal Information Updating

Shuai Li, *Member, IEEE*, Zhu-Hong You, *Member, IEEE*, Hongliang Guo, Xin Luo, *Member, IEEE*, and Zhong-Qiu Zhao

***Abstract*—The extreme learning machine (ELM) has drawn insensitive research attentions due to its effectiveness in solving many machine learning problems. However, the matrix inversion operation involved in the algorithm is computational prohibitive and limits the wide applications of ELM in many scenarios. To overcome this problem, in this paper, we propose an inverse-free ELM to incrementally increase the number of hidden nodes, and update the connection weights progressively and optimally. Theoretical analysis proves the monotonic decrease of the training error with the proposed updating procedure and also proves the optimality in every updating step. Extensive numerical experiments show the effectiveness and accuracy of the proposed algorithm.**

***Index Terms*—Extreme learning machine (ELM), inverse-free, neural networks, optimal updates.**

I. Introduction

IN PAST decades, neural networks, as powerful computational tools, have been extensively studied and successfully applied to solve various engineering problems [1]–[7] after the seminal work on the back-propagation (BP) learning rule [8], [9]. However, the previously overwhelming BP neural network, despite its advantage in approximating any nonlinearity under mild conditions [10], [11], demonstrates insufficiency in learning speed when exposed to datasets with a huge size which were never encountered tens of years ago. As an alternative to BP neural network, the extreme learning machine (ELM) is proposed to overcome the slow learning speed problem of feed-forward neural networks [12]. Different from the BP neural network which relies on the error propagation to compute the connection weights in iterations, the weights of ELM have an explicit and analytically expression, and can be solved efficiently using matrix pseudo-inversion. Since there is no iterative learning process needed, the slow learning speed problem of conventional feed-forward neural networks no longer holds for the ELM. The simplicity yet without loss of accuracy has casted the ELM with potentials to tackle big data learning problems and has endowed the ELM with quick development [13].

Some fundamental problems on ELMs, such as the universal approximation problem, the incremental learning problem, and the learning capacity problem [14], [15], have been explored and some theoretical conclusions have been drawn in recent years. It has been proved that ELMs have the ability to approximate any nonlinear multiple-input-multiple-output mapping with any desired accuracy [16]. Huang *et al.* [17] proposed an incremental algorithm to update the output weights incrementally with the increase of hidden node numbers. It is proved that the ELM using such an incremental algorithm is also an universal approximator. Zhang *et al.* [18] proposed an adaptive growth ELM (AG-ELM), which features adaptive determination of the required hidden nodes, incremental renewal of network weights, and sequential generation of a group new networks. It is proved in [18] that AG-ELM is able to approximate any nonlinearity under mild conditions. Zhang *et al.* [19] then improved the fundamental work on AG-ELM by proposing a dynamic ELM, which also bears universal approximation capability and is able to achieve a more compact network architecture than AG-ELM. In [20], the incremental ELM proposed in [17] is modified to an enhanced version, and is named as error minimization ELM (EM-ELM). Numerical experiments show an improved performance of EM-ELM over the incremental ELM. Feng *et al.* [21] introduced a new insight for efficient adjustments of ELM to remove insignificant hidden nodes. Based on this insight, a dynamic adjustment ELM is proposed by applying recursive expectation-minimization theorem to tune input parameters of insignificant hidden nodes for the reduction of residual errors. Besides the computational power in nonlinear regression, ELM is also applicable to nonlinear classification problems. The multiclass classification problem is investigated in [22] using the ELM. Due to the inherent connection between regression and classification, universal approximation theorem of the ELM for regression problems also implies its universal classification ability even in scenarios with complicated nonlinear classification boundaries. It has been shown in machine learning fields that the support vector machine (SVM) provides a powerful and robust linear classifier and can be efficiently applied to nonlinear classification problems by exploiting the

Manuscript received March 11, 2015; revised April 25, 2015; accepted May 13, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61401385, Grant 61373086, Grant 61202347, and Grant 61375047, and in part by the Young Scientist Foundation of Chongqing under Grant cstc2014kjrc-qnrc40005. This paper was recommended by Associate Editor G.-B. Huang. *(Corresponding author: Zhu-Hong You.)*

S. Li and Z.-H. You are with the Department Computing, Hong Kong Polytechnic University, Hong Kong.

H. Guo is with the School of Computer Engineering, Nanyang Technological University, Singapore.

X. Luo is with with Chongqing University, Chongqing, China.

Z.-Q. Zhao is with the Hefei University of Technology, Hefei, China (e-mail: zhuhong.you@polyu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2434841

2168-2267 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
COMPARISON OF THE PROPOSED ALGORITHM WITH EXISTING ELM ALGORITHMS

	Standard ELM [5]	Incremental ELM [10]	AG-ELM [11]	D-ELM [12]	EM-ELM [13]	DA-ELM [14]	The Proposed
Optimal	Yes	No	No	No	No	No	Yes
Universal Approximation	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Inverse Free	No	Yes	Yes	Yes	Yes	Yes	Yes
Incremental Update	No	Yes	Yes	Yes	Yes	Yes	Yes
Weight Error	Zero	Non-zero	Non-zero	Non-zero	Non-zero	Non-zero	Zero

kernel trick [23]. References [24]–[27] merged the SVM with the ELM by defining a new kernel and receive improved performances in both accuracy and speed. Most recently, it is found that stacking several ELMs together to a multilayered structure as an auto-encoder/decoder for deep learning can further improve the accuracy for nonlinear classification problems [28]. The seemingly contradiction between the simplicity in the structure of ELM and its powerfulness in solving regression and classification problems has evoked extensive researches, and the reason is still not fully unveiled. It is realized in [29] that the input layer setup of ELMs finds connections with the random projection algorithm [30], [31], which projects the original data into random directions as features for dimensionality reduction [32]. Motivated by this fact, some guidance on the selection of input layer weights in ELM, is adopted in [33] and an improved accuracy is observed in numerical experiments with such a guidance. Actually, not only in ELMs, the random weight selection technique is also adopted in some other models with great success, e.g., liquid state machines [34] and echo state networks [35], [36]. With the random assignments on the input weights, the output weights of ELMs can be directly determined by solving a linear least square problem, which bears similarities to designing the output weights of a radial basis function (RBF) network [37], [38] or neural networks with other activation functions [39], [40]. Besides, some other typical work on ELMs include using evolutionary strategy to optimize the input weights such that a compact ELM can be obtained [41], ELMs with sparse representation [42], integrating fuzzy logic with ELMs to improve the approximation performance [43], employing an ensemble of classifiers with the ELM as the base for performance improvement [44], applications of ELM for effective recognition of landmarks [45], [46], and insightful interpretation of ELMs from the perspective of random neurons, random features, and kernels [47].

Despite the great success of ELMs in superior learning accuracy and fast convergence, the involvement of an inverse operation in the calculation of the output weight may result in heavy computational overhead. Accordingly, the exploration of using inverse-free operators to determine the output weights of ELMs becomes promising. Early attempt along this line shows that universal approximation capability also can be reached by incrementally increasing the number of hidden nodes and updating weights in an inverse-free manner [17]. However, the obtained weights using the updating rule proposed in [17] is not the optimal one in the least square sense. In other words, the weights obtained using the method presented in [17] are not identical to the ones obtained using the conventional ELM algorithm, even under the same input weights, the same training set, and the same number of hidden nodes. From this perspective, the incremental strategy proposed in [17] computes output weights free of inverse operations at the cost of optimality. This evokes the following question: is there any way to obtain exact the same weights without using matrix inversion?

This paper makes progress along this direction and gives positive answer to this question (see Table I for a comparison of the proposed algorithm with existing ELM algorithms). With the assistance of the Sherman–Morrison formula and the Schur complement, we propose a strategy to incrementally update the output weights with the increase of the hidden node numbers. In every step, the obtained output weights are identical to the solution of the standard ELM algorithm using inverse operations. An immediate question following the incremental strategy to update the output weights in ELMs is: whether or not an improvement in the regression accuracy can be reached progressively in every step of increasing the hidden node numbers? It is noteworthy that this question cannot be answered by the universal approximation conclusion of ELMs as the universal approximation conclusion gives the ultimate learning capacity of ELMs while this question concerns on the learning progress in every step to increase the hidden node numbers. This question is answered theoretically in this paper by proving the monotonicity of the regression accuracy relative to the number of hidden nodes with the proposed method. The above mentioned twofold constructs the major contributions of this paper.

The reminder of this paper is organized as follows. In Section II, some preliminaries are given to support the theoretical derivation in this paper. The basic architecture of ELMs is briefed in Section III. In Section IV, the paradigm to increase the hidden node numbers, the inverse-free algorithm to update the output weights, and its extension to Tikhonov regularized version are given, respectively. Section V presents numerical validations on the proposed theoretical results by conducting extensive simulations. Section VI concludes this paper.

II. PRELIMINARIES

In this section, we present some useful preliminaries for the theoretical derivation in this paper. For $A \in \mathcal{R}^{m\times n}$ and $B \in \mathcal{R}^{p\times q}$, the Kronecker product $A \otimes B \in \mathcal{R}^{(mp)\times(nq)}$ is defined as the following block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}. \quad (1)$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

For an invertible square matrix A, and two vectors u and v of a proper size, the Sherman–Morrison formula states the following conclusion under the condition that $1 + v^T A^{-1} u \neq 0$:

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}. \tag{2}$$

The Schur complement provides a way to find matrix inverse via block decomposition. It gives the following equation when the referred matrix inverses all exist and the matrices are of proper sizes:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix} \tag{3}$$

with $E = (A - BD^{-1}C)^{-1}$, $F = -(A - BD^{-1}C)^{-1}BD^{-1}$, $G = -D^{-1}C(A - BD^{-1}C)^{-1}$, and $H = D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} + D^{-1}$.

III. Architecture of the ELM

With n input nodes, l hidden nodes and m output nodes, the ELM can be expressed in equation as follows:

$$\begin{aligned} h_i &= f\left(\sum_{k=1}^{n} a_{ik}x_k + b_i\right) \quad i = 1, 2, \ldots, l \\ z_j &= \sum_{i=1}^{m} w_{ji}h_i \quad j = 1, 2, \ldots, m \end{aligned} \tag{4}$$

where z_j is the value of the jth output node, h_i is the value of the ith hidden node, x_k is the kth input, $f(\cdot)$ is an activation function, which can be chosen as linear, sigmoid, Gaussian models, etc., a_{ik} is the input weights from the input node k to the hidden node i, b_i is the bias of the hidden node i, and w_{ji} is the output weight from the hidden node i to the output node j. Overall, the ELM model can be expressed in a compact form as follows:

$$\begin{aligned} h &= f(Ax + b) \\ z &= Wh \end{aligned} \tag{5}$$

where $z = [z_1, z_2, \ldots, z_m]^T \in \mathcal{R}^m$, $h = [h_1, h_2, \ldots, h_l]^T \in \mathcal{R}^l$, $x = [x_1, x_2, \ldots, x_n]^T \in \mathcal{R}^n$, $A = [a_{ik}] \in \mathcal{R}^{l\times n}$, $b = [b_1, b_2, \ldots, b_l]^T \in \mathcal{R}^l$, $W = [w_{ji}] \in \mathcal{R}^{m\times l}$, and $f(\cdot)$ is defined in entry-wise, i.e., $f(x) = [f(x_{ij})] \in \mathcal{R}^{m\times n}$ for a matrix input $x = [x_{ij}] \in \mathcal{R}^{m\times n}$. Specially, $f(x) = [f(x_1), f(x_2), \ldots, f(x_n)]$ for a vector input $x = [x_1, x_2, \ldots, x_n] \in \mathcal{R}^n$.

The ELM expression (4) can be regarded as a two-layered feed-forward neural network with the input layer choosing the activation function $f(\cdot)$ and the output layer using linear activation function without a bias. The way to solve network weights differs ELM from conventional neural networks the most. For conventional neural networks, the BP rule is commonly used to train the weights such that the desired output is achieved with given inputs. In contrast, ELM uses a completely different way to determine the weights, i.e., the value of A, b, and W in (5): A and b are chosen in random, and then W is accordingly obtained by minimizing the estimation error (the solution is a linear least-square solution to the problem). One may argue that the randomness of the input weights may result in nonoptimality in the obtained model. Actually, optimizing the input weights helps increase the approximation accuracy but is at the cost of robustness. Additionally, the universal approximation theorem of ELM guarantees to approximate any nonlinear piece continuous function using ELM with random input weights.

IV. Inverse-Free ELM

In practice, a small number of hidden nodes is preferable for computational considerations. However, the universal approximation theorem for ELM holds when the hidden node number approaches to infinity. This contrast motivates us to consider the tradeoff between the two factors: 1) computational effectiveness and 2) the approximation accuracy. One commonly used strategy in machine learning is to gradually increase the hidden node number until the desired accuracy is achieved. Although this strategy works well, it is at the cost of computing the weights for all neural networks with fewer hidden nodes. It is reasonable to guess that the weights may only change a little if only an extra hidden node is introduced, and there may exist a weight update law, which outputs new weights in the case of $n + 1$ hidden nodes based on the weight obtained in the case of n ones. In this part, we formalize this intuition and give rigorous equations for the update.

A. Proposed Model

We first define the hidden node increase policy of hidden nodes in the ELM to solve a given regression problem.

Definition 1 (Hidden Nodes Increase Policy): For a regression problem from $\mathcal{R}^n$ to $\mathcal{R}^m$, an ELM (5) with $l + 1$ ($l > 0$) hidden nodes is obtained by choosing the output weight W^{l+1} by the following standard linear least square algorithm used for ELM, and the input weights A^{l+1} and the bias b^{l+1} as:

$$A^{l+1} = \begin{bmatrix} A^l \\ \alpha^T \end{bmatrix}, \quad b^{l+1} = \begin{bmatrix} b^l \\ b_{l+1} \end{bmatrix} \tag{6}$$

where A^l and b^l is the input weights and biases for the same problem with l hidden nodes, $\alpha \in \mathcal{R}^n$ with entries chosen in the same probabilistic distribution as other elements in A^l, and $b_{l+1} \in \mathcal{R}$ chosen in the same probabilistic distribution as other elements in b^l.

Remark 1: The hidden nodes increase policy in Definition 1 differs from the incremental policy proposed in [17] in that they consider two different weight updating schemes. For the proposed one in this paper, the output weight is identical to the result obtained from standard ELM after weight updating by increasing a hidden node. In contrast, the new output weight following incremental policy in [17] after increasing a hidden node is not identical to the solution of standard ELM. Due to the optimality of standard ELM in the sense of least square training error, the node increase policy in Definition 1 implies its optimality.

Now we state the following theorem, which reveals the benefit of increasing the number of hidden nodes for approximation error.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Theorem 1: Using the hidden nodes increase policy in Definition 1, the training error of the resulting ELM decreases monotonically with the increase of the number of hidden nodes.

Proof: The approximation error of an ELM with given input weights A and biases b depends on its output weights. As to the ELM with $l+1$ hidden nodes, with given A^{l+1} and b^{l+1}, the approximation error depends on the output weight W^{l+1}. Denote x_i and y_i, the ith training input and the corresponding ith training output, respectively. Then, $y_1, y_2, \ldots, y_k \in \mathcal{R}^m$ is the desired output sequence in the training set, and $x_1, x_2, \ldots, x_k \in \mathcal{R}^n$ is the input sequence in the training set. In a compact form, the training input matrix is expressed as $X = [x_1, x_2, \ldots, x_k] \in \mathcal{R}^{n\times k}$, and the training output matrix is expressed as $Y = [y_1, y_2, \ldots, y_k] \in \mathcal{R}^{m\times k}$. According to the least square algorithm, $W^{l+1} = YB^{l+1}$ with $B^{l+1} = f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})(f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}))^{-1}$ (where k is the number of training examples, $\mathbf{1} \in \mathcal{R}^n$ is a vector with all entry 1.) is the optimal solution to the following problem:

$$\min_{W^{l+1}} \left\| Y - W^{l+1} f\left(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}\right)\right\|_F^2 \tag{7}$$

where $||\cdot||_F$ defines the Frobenius norm and $||Y - W^{l+1}f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})||_F^2 = \text{trace}((Y - W^{l+1}f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}))^T(Y - W^{l+1}f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})))$. Clearly, (7) is a quadratic programming problem and therefore the obtained optimal solution is indeed the global optimal one, meaning that the least square solution of W^{l+1} is the one with the lowest approximation error among all possible peers of W^{l+1}, including a particular one $W^{l+1} = [W^l \quad 0_{m\times 1}]$ with W^l being the least square output weight of the ELM with l hidden nodes. If we can further prove that this particular one has exactly the same approximation error as the ELM with l hidden nodes, we will be able to the draw the conclusion stated in this theorem. Actually, according to (6) and (7), for the ELM with such a particular choice on W^{l+1}, the approximation error equals $||Y - [W^l \quad 0_{m\times 1}]f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})||_F = ||Y - W^l f(A^l X + \mathbf{1}^T \otimes b^l)||_F$ which is identical to the approximation error for W^l in the case of the ELM with l hidden nodes. This completes the proof. ■

Remark 2: The universal approximation theorem for ELM states that any nonlinear piece continuous function can be approximated with any desired approximation error using an ELM. Without Theorem 1, one may suspect: is there any local optima in the number of hidden nodes such that locally increasing the hidden node number results in the increase of approximation errors? Theorem 1 gives a negative answer to this question and implies that one can keep increasing the hidden node number until the desired approximation error is reached.

Following the convention, we can compute W^{l+1}, which is associated with an ELM with $(l+1)$ hidden nodes, using the standard pseudo-inverse algorithm. However, this algorithm refers to the inverse of square matrices, which is usually computational heavy. The following theorem gives a solution to avoid using the inverse operator and to take advantage of the obtained weights of an ELM with l hidden nodes for the same approximation problem.

B. Algorithm

Theorem 2: For an ELM defined in Definition 1 with $l+1$ hidden nodes, its output weight $W^{l+1} = YB^{l+1}$ with $B^{l+1} = f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})(f(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}))^{-1}$ can be expressed using the output weights of the ELM with l hidden nodes in the following rule:

$$\begin{aligned} W^{l+1} &= YB^{l+1} \\ B^{l+1} &= \begin{bmatrix} B_1^{l+1} & B_2^{l+1} \end{bmatrix} \end{aligned} \tag{8}$$

with

$$\begin{aligned} B_1^{l+1} &= \frac{c^T c I - cc^T}{c^T c}\left(\frac{B^l H c c^T B^l}{c^T c - c^T B^l H c} + B^l\right) \\ B_2^{l+1} &= -\frac{B_1^{l+1} H c}{c^T c} + \frac{c}{c^T c} \end{aligned}$$

where $W^l = YB^l$ with $B^l = f^T(A^l X + \mathbf{1}^T \otimes b^l)(f(A^l X + \mathbf{1}^T \otimes b^l)f^T(A^l X + \mathbf{1}^T \otimes b^l))^{-1}$.

Proof: Substituting (6) into (8) yields

$$\begin{aligned} B^{l+1} = f^T\left(\begin{bmatrix} A^l X + \mathbf{1}^T \otimes b^l \\ \alpha^T X + b_{l+1}\mathbf{1}^T \end{bmatrix}\right)\Bigg(f\left(\begin{bmatrix} A^l X + \mathbf{1}^T \otimes b^l \\ \alpha^T X + b_{l+1}\mathbf{1}^T \end{bmatrix}\right) \\ \times f^T\left(\begin{bmatrix} A^l X + \mathbf{1}^T \otimes b^l \\ \alpha^T X + b_{l+1}\mathbf{1}^T \end{bmatrix}\right)\Bigg)^{-1}. \end{aligned} \tag{9}$$

Define $H = f(A^l X + \mathbf{1}^T \otimes b^l)$, $c = f(X^T\alpha + b_{l+1}\mathbf{1})$. Then, noting that $f(\cdot)$ is a mapping operated in entry-wise, (9) equivalently writes

$$\begin{aligned} B^{l+1} &= \begin{bmatrix} H \\ c^T \end{bmatrix}^T \left(\begin{bmatrix} H \\ c^T \end{bmatrix}\begin{bmatrix} H \\ c^T \end{bmatrix}^T\right)^{-1} \\ &= \begin{bmatrix} H^T & c \end{bmatrix}\begin{bmatrix} HH^T & Hc \\ c^T H^T & c^T c \end{bmatrix}^{-1}. \end{aligned} \tag{10}$$

Using Schur complement, we obtain

$$\begin{bmatrix} HH^T & Hc \\ c^T H^T & c^T c \end{bmatrix}^{-1} = \begin{bmatrix} E_0 & F_0 \\ F_0^T & G_0 \end{bmatrix} \tag{11}$$

with

$$E_0 = \left(HH^T - \frac{Hcc^T H^T}{c^T c}\right)^{-1} \tag{12}$$

$$F_0 = -\frac{E_0 Hc}{c^T c} \tag{13}$$

$$G_0 = \frac{c^T H^T E_0 Hc}{\left(c^T c\right)^2} + \frac{1}{c^T c}. \tag{14}$$

Therefore, we have

$$B^{l+1} = \begin{bmatrix} H^T E_0 + cF_0^T & H^T F_0 + cG_0 \end{bmatrix}. \tag{15}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

As to E_0 expressed in (12), supposing that $c^Tc - c^TH^T(HH^T)^{-1}Hc \neq 0$, it can be transformed as follows using the Sherman–Morrison formula:

$$\begin{aligned} E_0 &= \left(HH^T - \frac{Hcc^TH^T}{c^Tc}\right)^{-1} \\ &= \frac{\left(HH^T\right)^{-1}Hcc^TH^T\left(HH^T\right)^{-1}}{c^Tc - c^TH^T\left(HH^T\right)^{-1}Hc} + \left(HH^T\right)^{-1}. \end{aligned} \tag{16}$$

On the other hand, the output weight W^l of the ELM with l hidden nodes for the same approximation problem writes as follows:

$$\begin{aligned} W^l &= YB^l \\ B^l &= f^T\left(A^lX + \mathbf{1}^T \otimes b^l\right)\left(f\left(A^lX + \mathbf{1}^T \otimes b^l\right)\right. \\ &\qquad \left. \times f^T\left(A^lX + \mathbf{1}^T \otimes b^l\right)\right)^{-1} \\ &= H^T\left(HH^T\right)^{-1}. \end{aligned} \tag{17}$$

The expression of B^{l+1} can be obtained by substituting (12)–(14) into (15). With the assistance of (17), B^{l+1} can be simplified to the following:

$$\begin{aligned} H^TE_0 + cF_0^T &= H^TE_0 - c\left(\frac{E_0Hc}{c^Tc}\right)^T \\ &= \left(I - \frac{cc^T}{c^Tc}\right)H^TE_0 \\ &= \frac{c^TcI - cc^T}{c^Tc}\left(\frac{B^lHcc^TB^l}{c^Tc - c^TB^lHc} + B^l\right) \end{aligned} \tag{18}$$

$$\begin{aligned} H^TF_0 + cG_0 &= -\frac{H^TE_0Hc}{c^Tc} + \frac{c\left(c^TH^TE_0\right)Hc}{\left(c^Tc\right)^2} + \frac{c}{c^Tc} \\ &= -\frac{H^TE_0Hc}{c^Tc} - \frac{cF_0^THc}{c^Tc} + \frac{c}{c^Tc} \\ &= -\frac{\left(H^TE_0 + cF_0^T\right)Hc}{c^Tc} + \frac{c}{c^Tc}. \end{aligned} \tag{19}$$

Notice that $-(c^TH^TE_0)/(c^Tc) = F_0^T$ since $E_0 = E_0^T$. Equations (18) and (19) together construct the two components of B^{l+1} and complete the proof. ■

Remark 3: For the case without Tikhonov regularization, the proof of Theorem 2 relies on the assumption that HH^T is nonsingular and c^Tc is nonzero. The restriction $c^Tc - c^TH^T(HH^T)^{-1}Hc \neq 0$ referred in the proof is implied by the assumptions on the nonsingularity of HH^T and the nonzeroness of c^Tc, plus that the standard ELM output weight W^{l+1} exists. For the case with Tikhonov regularization, the items HH^T and c^Tc are replaced by $HH^T + k_0\mathbf{I}$, which is always positive definite and nonsingular for $k_0 > 0$, and $c^Tc + k_0 > 0$. In this situation, the restrictions are directly satisfied.

Remark 4: For regression problems with strong nonlinearity or with a high dimension, it is often necessary to use many hidden nodes (say the number of hidden nodes is l) to reach a satisfactory approximation accuracy. A resulting problem is the heavy computation burden to calculate the inverse of a $l \times l$ matrix using conventional algorithms for ELM. With Theorem 2, there is no need to compute any matrix inverses and the computational expenses are thoroughly reduced.

Algorithm 1 Least Square ELM by Incrementally Increasing the Number of Hidden Nodes

Require:

Desired approximation error η, input dimension n, output dimension m, training set size k, the training input $X = [x_1, x_2, \ldots, x_k] \in \mathcal{R}^{n\times k}$, the training output $Y = [y_1, y_2, \ldots, y_k] \in \mathcal{R}^{m\times k}$, the initial ELM model with l_0 hidden nodes (input weight $A^{l_0} \in \mathcal{R}^{l_0\times n}$, input bias $b^{l_0} \in \mathcal{R}^{l_0}$, pseudo-inverse $B^{l_0} \in \mathcal{R}^{k\times l_0}$, output weight $W^{l_0} \in \mathcal{R}^{m\times n}$).

Ensure:

An ELM with l hidden nodes (A, b and W) such that the approximation error $\eta*$ is reached.

1: $l = l_0$, $A = A^{l_0}$, $b = b^{l_0}$, $W = W^{l_0}$
2: $H = f(AX + \mathbf{1}^T \otimes b)$
3: $E = Y - WH$ //initial training error
4: $\eta = MSE(E)$ //initial mean square error
5: **while** $(\eta > \eta*)$ **do**
6: $\alpha \Leftarrow$ a random vector in $\mathcal{R}^n$
7: $A \Leftarrow \begin{bmatrix} A \\ \alpha^T \end{bmatrix}$ //update A
8: $b_0 \Leftarrow$ a random scalar
9: $b \Leftarrow \begin{bmatrix} b \\ b_0 \end{bmatrix}$ //update b
10: $H = f(AX + \mathbf{1}^T \otimes b)$, $c = f(X^T\alpha + b_0\mathbf{1})$
11: $B_1 \Leftarrow \frac{c^TcI - cc^T}{c^Tc}\left(\frac{BHcc^TB}{c^Tc - c^TBHc} + B\right)$
12: $B_2 \Leftarrow -\frac{B_1Hc}{c^Tc} + \frac{c}{c^Tc}$
13: $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$
14: $W = YB$ //update W
15: $E = Y - WH$ //training error
16: $\eta = MSE(E)$ //mean square error
17: **end while**

Based on Theorem 2, regression problems using an ELM with the desired approximation error $\eta*$ ($\eta* > 0$) can be solved by incrementally increasing the number of hidden nodes according to Algorithm 1.

C. Extension to ELM With Tikhonov Regularization

Tikhonov regularization is commonly used in machine learning to avoid over-fitting [48], [49]. As to the cases with l and $l+1$ hidden nodes, respectively, the output weights for ELMs with Tikhonov regularization write as follows:

$$\begin{aligned} W^l &= YB^l \\ B^l &= f^T\left(A^lX + \mathbf{1}^T \otimes b^l\right)\left(k_0^2I + f\left(A^lX + \mathbf{1}^T \otimes b^l\right)\right. \\ &\qquad \left. \times f^T\left(A^lX + \mathbf{1}^T \otimes b^l\right)\right)^{-1} \end{aligned} \tag{20}$$

$$\begin{aligned} W^{l+1} &= YB^{l+1} \\ B^{l+1} &= f^T\left(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}\right) \\ &\quad \times \left(k_0^2I + f\left(A^{l+1}X + \mathbf{1}^T \otimes b^l\right)\right. \\ &\qquad \left. \times f^T\left(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}\right)\right)^{-1}. \end{aligned} \tag{21}$$

For regularized ELMs, the weight updating law is stated in the following theorem.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Theorem 3: For ELMs defined in Definition 1 with the Tikhonov regularization, the output weight for the ELM with $l+1$ hidden nodes, as expressed in (21), can be expressed using the output weights of the ELM with l hidden nodes as expressed in (20), using the updating equation (8), with B_1^{l+1} and B_2^{l+1} in (8) defined as

$$B_1^{l+1} = \frac{\left((c^Tc+k_0^2)I - cc^T\right)B^lHcc^TB^l}{\left(c^Tc+k_0^2\right)\left(c^Tc+k_0^2-c^TB^lHc\right)} + \frac{\left((c^Tc+k_0^2)I-cc^T\right)B^l}{c^Tc+k_0^2}$$
$$B_2^{l+1} = -\frac{B_1^{l+1}Hc}{c^Tc+k_0^2} + \frac{c}{c^Tc+k_0^2}. \tag{22}$$

Proof: For statement convenience, we define $H = f(A^lX + \mathbf{1}^T \otimes b^l)$, $c = f(X^T\alpha + b_{l+1}\mathbf{1})$. Introduce the following auxiliary matrix:

$$\bar{f}_l = \begin{bmatrix} f\left(A^lX + \mathbf{1}^T \otimes b^l\right) & k_0I \end{bmatrix} = \begin{bmatrix} H & k_0I \end{bmatrix}. \tag{23}$$

Then, its pseudo-inverse $\bar{B}^l$ is

$$\bar{B}^l = \bar{f}_l^T\left(\bar{f}_l\bar{f}_l^T\right)^{-1} = \begin{bmatrix} H^T\left(k_0^2I+HH^T\right)^{-1} \\ k_0\left(k_0^2I+HH^T\right)^{-1} \end{bmatrix} \tag{24}$$

where $\bar{B}^l$ is a $(l+k)\times l$-dimensional matrix (recall that A^l is $l\times n$-dimensional and X is $n\times k$-dimensional, therefore $\bar{f}$ is $l\times(l+k)$-dimensional), $H^T(k_0^2I+HH^T)^{-1}$ is $k\times l$-dimensional and $k_0(k_0^2I+HH^T)^{-1}$ is $l\times l$-dimensional. Clearly, the first k rows of $\bar{B}^l$ is the solution of B^l with the Tikhonov regularization coefficient k_0 or in

$$B^l = \begin{bmatrix} I_{k\times k} & 0_{k\times l} \end{bmatrix}\bar{B}^l. \tag{25}$$

Similarly, for ELM with $l+1$ hidden nodes

$$B^{l+1} = \begin{bmatrix} I_{k\times k} & 0_{k\times(l+1)} \end{bmatrix}\bar{B}^{l+1}. \tag{26}$$

For $\bar{f}$ in the case of $l+1$ hidden nodes, we have

$$\bar{f}_{l+1} = \begin{bmatrix} H & k_0I \end{bmatrix} = \left[\begin{array}{c|cc} H & k_0I & 0 \\ c^T & 0 & k_0 \end{array}\right]. \tag{27}$$

Following the same procedure as in the proof of Theorem 2, the recursive updating rule for $\bar{B}^{l+1}$ writes:

$$\bar{B}^{l+1} = \begin{bmatrix} \bar{B}_1^{l+1} & \bar{B}_2^{l+1} \end{bmatrix} \tag{28}$$

with

$$\bar{B}_1^{l+1} = \frac{c_1^Tc_1I - c_1c_1^T}{c_1^Tc_1}\left(\frac{\bar{B}'H_1c_1c_1^T\bar{B}'}{c_1^Tc_1 - c_1^T\bar{B}'H_1c_1} + \bar{B}'\right)$$
$$\bar{B}_2^{l+1} = -\frac{\bar{B}_1^{l+1}H_1c_1}{c_1^Tc_1} + \frac{c_1}{c_1^Tc_1} \tag{29}$$

where

$$H_1 = \begin{bmatrix} H & k_0I & 0_{l\times 1} \end{bmatrix}, c_1 = \begin{bmatrix} c \\ 0_{l\times 1} \\ k_0 \end{bmatrix}, \bar{B}' = \begin{bmatrix} \bar{B}^l \\ 0_{1\times l} \end{bmatrix}. \tag{30}$$

With (24) and (30), we obtain, $H_1c_1 = Hc$, $[I_{k\times k} \quad 0_{k\times(l+1)}](c_1c_1^T - c_1c_1^T) = [(c^Tc+k_0^2)I - cc^T \; 0 \; -k_0c]$, $[I_{k\times k} \quad 0_{k\times(l+1)}]\ (c_1c_1^T - c_1c_1^T)\bar{B}' = (c^Tc+k_0^2)I - cc^T)H^T(k_0^2I+HH^T)^{-1} = (c^Tc+k_0^2)I - cc^T)B^l$, $c_1^T\bar{B}' = c^TH^T(k_0^2I+HH^T)^{-1} = c^TB^l$, also we have

$$\bar{B}'H_1c_1 = \begin{bmatrix} H^T\left(k_0^2I+HH^T\right)^{-1}Hc \\ k_0\left(k_0^2I+HH^T\right)^{-1}Hc \\ 0 \end{bmatrix}$$

and $c_1^T\bar{B}'H_1c_1 = c^TH^T(k_0^2I+HH^T)^{-1}Hc = c^TB^lHc$, $[I_{k\times k} \quad 0_{k\times(l+1)}](c_1c_1^T - c_1c_1^T)\bar{B}'H_1c_1 = ((c^Tc+k_0^2)I - cc^T)H^T(k_0^2I+HH^T)^{-1}Hc = ((c^Tc+k_0^2)I - cc^T)B^lHc$, $[I_{k\times k} \quad 0_{k\times(l+1)}](c_1c_1^T - c_1c_1^T)\bar{B}'H_1c_1c_1^T\bar{B}' = ((c^Tc+k_0^2)I - cc^T)H^T(k_0^2I+HH^T)^{-1}Hcc^TH^T(k_0^2I+HH^T)^{-1} = ((c^Tc+k_0^2)I - cc^T)B^lHcc^TB^l$

$$\begin{bmatrix} I_{k\times k} & 0_{k\times(l+1)} \end{bmatrix}\bar{B}_1^{l+1} = \frac{\left((c^Tc+k_0^2)I - cc^T\right)B^lHcc^TB^l}{\left(c^Tc+k_0^2\right)\left(c^Tc+k_0^2-c^TB^lHc\right)} + \frac{\left((c^Tc+k_0^2)I-cc^T\right)B^l}{c^Tc+k_0^2} \tag{31}$$

$$\begin{bmatrix} I_{k\times k} & 0_{k\times(l+1)} \end{bmatrix}\bar{B}_2^{l+1} = -\frac{\begin{bmatrix} I_{k\times k} & 0_{k\times(l+1)} \end{bmatrix}\bar{B}_1^{l+1}Hc}{c^Tc+k_0^2} + \frac{c}{c^Tc+k_0^2} \tag{32}$$

which constructs the components for B^{l+1} according to (26) and (28). ■

Based on Theorem III, a Tikhonov regularized ELM for regression problems with the desired approximation error $\eta*$ ($\eta* > 0$) can be solved by incrementally increasing the number of hidden nodes according to Algorithm 2.

D. Complexity Analysis

In this part, we give analysis on the complexity of the proposed algorithms. Since the result presented in (Section IV-B) can be regarded as a special case of the Tikhonov regularized least square ELM when choosing the regularization coefficient $k_0 = 0$. Without losing generality, we only analyze the time complexity of Algorithm 2.

For each iteration (lines 5–18) in Algorithm 2, lines 12, 13, and 15 dominate the computational burdens. Recall that the multiplication of two matrices, one of size $l_1 \times l_2$ and the other one of size $l_2 \times l_3$ requires $l_1l_3(2l_2-1)$ flops of operation and the summation of two matrices in size $l_1 \times l_2$ requires l_1l_2 flops of operations [50]. When there are l hidden nodes, the dimension of the matrices B, H, Y, and c are $k \times l$, $l \times k$, $m \times k$, and $k \times 1$, respectively. For line 12, we first reorganize $((c^Tc+k_0^2)I - cc^T)BHcc^TB$ as $((c^Tc+k_0^2)I - cc^T)BHcc^TB = (c^Tc+k_0^2)(BHc)(c^TB) - c(c^T(BHc))(c^TB)$ (note that c^TBHc is a scalar). For this quantity, $c^Tc+k_0^2$, c^TB, and Hc need $2k+1$, $l(2k-1)$, and $l(2k-1)$ flops of operations, respectively. The computation of BHc based on the result of Hc needs $k(2l-1)$ flops. Further $c^T(BHc)$ and $(BHc)(c^TB)$ need $2k-1$ and kl flops, respectively. Then, the computation of $c(c^T(BHc))(c^TB)$ costs $kl+k$ additional flops for computation. In addition, $(c^Tc+k_0^2)(BHc)(c^TB)$ needs $kl+k$ computational flops. Overall, $((c^Tc+k_0^2)I-cc^T)BHcc^TB$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 2 Tikhonov Regularized Least Square ELM by Incrementally Increasing the Number of Hidden Nodes

Require:

Desired approximation error η, regularization factor k_0, input dimension n, output dimension m, training set size k, the training input $X = [x_1, x_2, \ldots, x_k] \in \mathcal{R}^{n\times k}$, the training output $Y = [y_1, y_2, \ldots, y_k] \in \mathcal{R}^{m\times k}$, the initial ELM model with l_0 hidden nodes (input weight $A^{l_0} \in \mathcal{R}^{l_0\times n}$, input bias $b^{l_0} \in \mathcal{R}^{l_0}$, regularized pseudo-inverse $B^{l_0} \in \mathcal{R}^{k\times l_0}$, output weight $W^{l_0} \in \mathcal{R}^{m\times n}$).

Ensure:

An ELM with l hidden nodes (A, b and W) such that the approximation error $\eta*$ is reached.

1: $l = l_0$, $A = A^{l_0}$, $b = b^{l_0}$, $B = B^{l_0}$, $W = W^{l_0}$
2: $H = f(AX + \mathbf{1}^T \otimes b)$
3: $E = Y - WH$ //initial training error
4: $\eta = MSE(E)$ //initial mean square error
5: **while** $(\eta > \eta*)$ **do**
6: $l \Leftarrow l + 1$
7: $\alpha \Leftarrow$ a random vector in $\mathcal{R}^n$
8: $A \Leftarrow \begin{bmatrix} A \\ \alpha^T \end{bmatrix}$ //update A
9: $b_0 \Leftarrow$ a random scalar
10: $b \Leftarrow \begin{bmatrix} b \\ b_0 \end{bmatrix}$ //update b
11: $H = f(AX + \mathbf{1}^T \otimes b)$, $c = f(X^T\alpha + b_0\mathbf{1})$
12: $B_1 \Leftarrow \frac{((c^Tc+k_0^2)I-cc^T)BHcc^TB}{(c^Tc+k_0^2)(c^Tc+k_0^2-c^TBHc)} + \frac{((c^Tc+k_0^2)I-cc^T)B}{c^Tc+k_0^2}$
13: $B_2 \Leftarrow -\frac{B_1Hc}{c^Tc+k_0^2} + \frac{c}{c^Tc+k_0^2}$
14: $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$
15: $W = YB$ //update W
16: $E = Y - WH$ //training error
17: $\eta = MSE(E)$ //mean square error
18: **end while**

needs $9kl + 5k - 2l$ flops. Since the values of $c^Tc + k_0^2$ and c^TBHc have been obtained until now, it only needs 2 additional flops to compute $(c^Tc + k_0^2)(c^Tc + k_0^2 - c^TBHc)$. As to $((c^Tc + k_0^2)I - cc^T)B = (c^Tc + k_0^2)B - cc^TB$, a total of $2kl + 1$ flops are needed to conduct the computation. In total, line 12 needs $14kl + 5k - 2l + 3$ flops to assign B_1 with the updated value. For line 13, we need an additional $2kl + 2k$ flops to update the value of B_2. Line 15 costs $m(l+1)(2k-1)$ flops for computation. All together, lines 12, 13, and 15 need $m(l + 1)(2k - 1) + 16kl + 7k - 2l + 3$ flops of computation, which is approximately $(2m + 16)kl$ for large k (the number of training examples) and l (the number of hidden nodes), or $O(kl)$ in the sense of time complexity.

The conventional Tikhonov regularized ELM replaces lines 12 and 13 in Algorithm 2 with (21). Notice that $f(A^{l+1}X+\mathbf{1}^T\otimes b^{l+1})$ is $(l+1)\times k$-dimensional. It takes $2(l+1)+(2k-1)(l+1)^2$ flops to calculate $k_0^2I+f(A^{l+1}X+\mathbf{1}^T\otimes b^l)f^T(A^{l+1}X+\mathbf{1}^T\otimes b^{l+1})$. The operation of matrix inversion has a time complexity of $O(n^3)$ for a $n \times n$ matrix using Gaussian–Jordan elimination. Therefore, the time complexity of is $O(l^3)$ to calculate $(k_0^2I+f(A^{l+1}X+\mathbf{1}^T\otimes b^l)f^T(A^{l+1}X+\mathbf{1}^T\otimes b^{l+1}))^{-1}$. Together with the additional expenses of $k(l + 1)(2l + 1)$ to compute $f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1})(k_0^2I + f(A^{l+1}X + \mathbf{1}^T \otimes b^l)f^T(A^{l+1}X + \mathbf{1}^T \otimes b^{l+1}))^{-1}$, the overall time complexity is $O(l^3 + kl^2)$ for the conventional Tikhonov regularized ELM. Comparing the complexity of $O(l^3 + kl^2)$ for the conventional algorithm while $O(kl)$ for the proposed one, it can be concluded that the proposed algorithm significantly reduces computational expenses.

TABLE II
PROPERTIES OF BENCHMARK REGRESSION DATASETS

datasets	Samples	Features	Randomness
Airfoil self-noise	1503	6	Yes
Energy efficiency	768	8	Yes
Housing	506	14	No
Parkinson	5875	22	Yes
Protein	45730	9	Yes

E. Discussion

In the previous sections, we have developed the inverse-free ELM weight updating rules to solve regression problems. Due to the inherent connections between regression and classification problems, the same framework also applies to classification problems. For a m-class classification problem, the desired output for a training sample belonging to class i can be constructed as $y = e_i$, where $e_i \in \mathcal{R}^m$ is a vector having all elements zero except that the ith one equal to 1. The training set then can be used to train the ELM by following the same procedure as for the regression problem. In the prediction phase, class labels of the testing data are obtained by identifying the largest element in the m-dimensional output vector generated by the ELM fed with test inputs. For example, in the case that the output has the largest value in the ith element among all m output elements, the sample is classified into class i. With this formulation, the proposed algorithm applies to both the regression and the classification problems.

V. NUMERICAL EXPERIMENTS

This section conducts numerical experiments to validate the effectiveness of the proposed inverse-free ELM algorithm by applying it to real-world benchmark regression and classification problems.

A. Benchmark Datasets

In order to extensively evaluate the performance of the proposed algorithm, we consider several representative scenarios which are often encountered in machine learning fields, namely large sizes, small sizes, high dimensions, and/or low dimensions. The experimental datasets consist of five classification cases and five regression cases, with the datasets coming from the University of California, Irvine machine learning repository [51] and library for support vector machines dataset [52].

In specific, for the regression problem, we consider datasets as summarized in Table II: energy efficiency dataset, which is of small size and low dimensions [53]; housing dataset, which is of small size and medium dimensions [54]; Parkinson disease dataset, which is of medium size and medium dimensions [55]; airfoil self-noise dataset, which is of medium

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
PROPERTIES OF BENCHMARK REGRESSION AND CLASSIFICATION DATASETS

datasets	Samples	Features	Randomness
Diabetes	768	8	Yes
Leukemia	72	7129	No
MAGIC Gamma telescope	19020	10	Yes
Musk	6598	167	Yes
Adult	30956	123	Yes

size and low dimensions [56]; physicochemical properties of protein dataset, which is of large size and low dimensions [51]. In Table II, column "randomness" represents whether the training and testing data of the corresponding datasets are reshuffled at each trial of simulation. If the training and testing data of the datasets remain fixed for all trials of simulations, it is marked "no." Otherwise, it is marked "yes." Preprocessing is carried out for training data, making all attributes linearly scaled into $[-1, 1]$. Attributes of testing data are also scaled accordingly based on factors used in the scaling of training data.

For the classification problem, we consider datasets as summarized in Table III: diabetes dataset, which is of small size and low dimensions [51]; musk dataset, which is of medium size and medium dimensions [57]; MAGIC Gamma telescope dataset, which is of large size and low dimensions [58]; leukemia dataset, which is of small size and high dimensions [59]; adult dataset, which is of large size and medium dimensions [60].

B. Simulation Environment Settings

In the study, the simulations of different algorithms on all the datasets are evaluated with MATLAB R2011a environment running on an Intel i5-2400 3.10 GHz CPU with 6.00 GB RAM. To guarantee the experimental results are valid and can be generalized for making predictions regarding new data, each benchmark dataset is randomly partitioned into training and independent testing sets via a fivefold cross validation. Each of the five subsets acts as an independent holdout testing dataset for the model trained with the rest of four subsets. Thus, five models were generated for the five sets of data. Their mean accuracy and variance are computed for comparison. The advantages of cross validation are that the impact of data dependency is minimized and the reliability of the results can be improved.

C. Evaluation Metrics

For the classification problem, we employ four commonly used indices as the measure for performance: the overall prediction accuracy (ACC), defined as $\text{ACC} = (\text{TP} + \text{TN}/\text{TP} + \text{FP} + \text{TN} + \text{FN})$ with TP, FN, FP and TN denoting true positive, false negative, false positive, and true negative; the sensitivity (SN), defined as $\text{SN} = (\text{TP}/\text{TP} + \text{FN})$; the precision (PE) defined as $\text{PE} = (\text{TP}/\text{TP} + \text{FP})$ and the Matthews correlation coefficient (MCC) defined as $\text{MCC} = (\text{TP} \times \text{TN} - \text{FP} \times \text{FN}/\sqrt{(\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TP} + \text{FP}) \times (\text{TN} + \text{FN})})$. For the regression problem, the mean squared error (MSE), the error variance are employed to evaluate the prediction performance of the proposed model.

D. Experimental Results

Table IV shows the regression performance of the proposed inverse-free ELM with different activation functions, including the sigmoidal function, the sine function, the hardlim function, the triangular function, and the RBF. In the simulations, we use the proposed algorithm with Tikhonov regularization using $k_0 = 0.1$ to avoid over-fitting. Compared with conventional ELM using the same regularization factor, as observed from Table IV, inverse-free ELM results in the same solutions on both the connection weight matrix and the output for the same sample inputs by observing that the weight error (the weight error is defined as $\|W_1 - W_2\|_F$ with W_1 and W_2 being the weights obtained by the proposed algorithm and the standard ELM algorithm, respectively), which is smaller than 10^{-10} for all tested datasets, and the output error [the output error is defined as the difference between the nominal output and the predicted output. For the output error of the training set and that of the testing set, we refer them as output error (training) and output error (testing) in Table IV, respectively.], which is less than 10^{-13} (the output error is defined as $\|Y_1 - Y_2\|_F$ with Y_1 and Y_2 being the outputs obtained by the proposed algorithm and the standard ELM algorithm). Take the airfoil self-noise and the physicochemical properties of protein datasets in Table IV as examples. For the airfoil self-noise dataset, the training and testing MSE values of inverse-free ELM with Gaussian kernel are 0.0304 and 0.0303, respectively. The training and the testing output difference between the standard ELM and the proposed algorithm are 3.22×10^{-14} and 1.75×10^{-16}, respectively, which is ignorably small taking into account the computation error. For physicochemical properties of protein dataset, the training and testing MSE values of inverse-free ELM with Gaussian kernel are 0.0734 and 0.0735, respectively. The training and testing output difference between the standard ELM and the proposed inverse-free ELM are 2.89×10^{-15} and 2.43×10^{-15}, respectively. In conclusion, although the ELM reaches different accuracy for different datasets with different kernel selections, the proposed algorithm always reaches the same accuracy as the standard ELM algorithm, as observable from Fig. 1. Overall, the extensive experimental results validate the fact that the proposed inverse-free ELM reaches the same result as the standard ELM in solving regression problems.

In this part, we report the experimental results of the proposed inverse-free scheme in solving various classification problems. Table V demonstrates the classification performance comparison of ELM and the proposed inverse-free ELM with different activation functions. We can see from Table V that the inverse-free ELM of kernel form can always achieves comparable performance as ELM for most datasets. Take adult dataset (large number of training samples) as an example. It can be observed from Table V that the proposed inverse-free ELM with Gaussian kernel can achieve a relatively high prediction accuracy of 84.28%. To better investigate the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV
EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHM TO SOLVE REGRESSION PROBLEMS

Dataset	Kernel	Weight error	Output error (training)	Output error (testing)	Training		Testing	
					MSE	Variance	MSE	Variance
Airfoil	Gaussian	3.22E-14	1.03E-16	1.75E-16	0.0304	0.0046	0.0303	0.0057
	Sigmoid	7.35E-14	1.87E-16	1.92E-16	0.0285	0.0045	0.0280	0.0044
	Hardlim	9.79E-15	3.47E-17	2.08E-17	0.0321	0.0049	0.0317	0.0039
	Triangular	1.40E-13	2.64E-17	9.02E-17	0.0337	0.0048	0.0345	0.0058
	Sine	1.14E-14	1.11E-17	4.09E-17	0.0308	0.0041	0.0310	0.0051
Energy	Gaussian	3.96E-15	1.60E-17	1.77E-17	0.0308	0.0103	0.0303	0.0110
	Sigmoid	2.11E-14	1.41E-16	7.95E-17	0.0307	0.0120	0.0321	0.0146
	Hardlim	2.19E-14	1.14E-16	6.31E-17	0.0437	0.0065	0.0426	0.0114
	Triangular	3.80E-15	2.98E-17	2.81E-17	0.0269	0.0140	0.0272	0.0182
	Sine	4.11E-15	1.14E-17	1.94E-17	0.0192	0.0076	0.0196	0.0089
Housing	Gaussian	2.78E-15	1.39E-17	5.00E-17	0.0298	0.0062	0.0293	0.0075
	Sigmoid	7.02E-15	7.98E-17	4.65E-17	0.0311	0.0056	0.0315	0.0081
	Hardlim	1.44E-15	5.55E-18	2.29E-17	0.0386	0.0054	0.0382	0.0072
	Triangular	1.07E-15	6.94E-18	1.25E-17	0.0435	0.0064	0.0460	0.0138
	Sine	1.95E-15	1.39E-17	2.64E-17	0.0305	0.0072	0.0312	0.0086
Parkinson	Gaussian	1.63E-14	3.12E-17	2.15E-17	0.0404	0.0087	0.0416	0.0095
	Sigmoid	5.48E-13	1.77E-15	2.03E-15	0.0363	0.0055	0.0364	0.0056
	Hardlim	2.58E-13	9.85E-17	2.53E-16	0.0491	0.0009	0.0494	0.0024
	Triangular	9.37E-15	1.80E-17	4.72E-17	0.0468	0.0017	0.0461	0.0045
	Sine	2.37E-14	3.47E-17	3.05E-17	0.0389	0.0073	0.0391	0.0075
Protein	Gaussian	3.49E-12	2.89E-15	2.43E-15	0.0734	0.0055	0.0735	0.0053
	Sigmoid	3.38E-11	1.50E-14	1.32E-14	0.0741	0.0043	0.0743	0.0037
	Hardlim	2.76E-10	2.35E-15	5.69E-15	0.0790	0.0009	0.0791	0.0015
	Triangular	3.04E-13	1.94E-16	1.69E-16	0.0741	0.0043	0.0744	0.0050
	Sine	1.70E-12	1.55E-15	1.69E-15	0.0735	0.0036	0.0735	0.0038

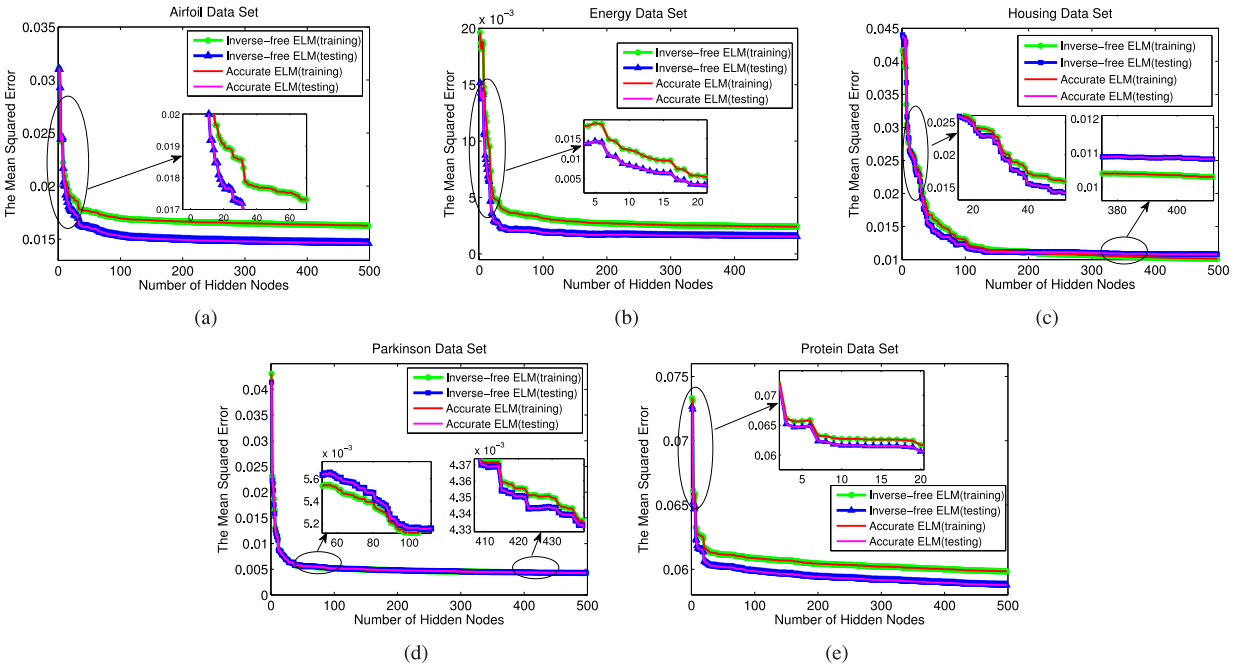


Fig. 1. MSEs for the proposed inverse-free ELM and the standard matrix inversion-based accurate ELM. (a) Airfoil self-noise dataset. (b) Energy efficiency dataset. (c) Housing dataset. (d) Parkinson dataset. (e) Physicochemical properties of protein dataset.

prediction ability of our model, we also calculated the values of SN, PE, and MCC. From Table V, we can see that our model gives good prediction performance with an average SN value of 54.24%, PE value of 73.46%, and MCC value of 60.07%. Further, it can also be seen in Table V that the variance of SN, PE, ACC, MCC, and area under curve are as low as 0.0126, 0.0064, and 0.0078, respectively. The receiver operator characteristic (ROC) curve, which plots the achievable sensitivity (the size of true positives that can be detected by our method) at a given specificity (one-false positive rate) is commonly used to evaluate the performance of a classification algorithm. As shown in Fig. 2, a stronger bend toward the upper-left corner of the ROC graph (i.e., high sensitivity is achieved with a low-false positive rate) can be found with the proposed algorithm, indicating that the proposed inverse-free ELM can successfully classify positive and negative samples with all five activation functions that we investigated. To sum up, it is clear that the ELM algorithm reaches satisfactory results over the test

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE V
EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHM TO SOLVE CLASSIFICATION PROBLEMS

Dataset	Kernel	Mean/Variance	Training				Testing			
			ACC	SN	PE	MCC	ACC	SN	PE	MCC
MAGIC	Gaussian	Mean	0.8698	0.9447	0.8666	0.7407	0.8586	0.9362	0.8586	0.7218
		Variance	0.0030	0.0019	0.0027	0.0049	0.0056	0.0039	0.0050	0.0105
	Sigmoid	Mean	0.8685	0.9498	0.8616	0.7368	0.8656	0.9484	0.8589	0.7315
		Variance	0.0027	0.0020	0.0029	0.0044	0.0061	0.0035	0.0069	0.0087
	Hardlim	Mean	0.8569	0.9323	0.8590	0.7198	0.8457	0.9248	0.8504	0.7013
		Variance	0.0022	0.0017	0.0023	0.0046	0.0046	0.0055	0.0048	0.0099
	Triangular	Mean	0.8596	0.9359	0.8599	0.7238	0.8441	0.9228	0.8498	0.6991
		Variance	0.0022	0.0016	0.0024	0.0034	0.0074	0.0060	0.0069	0.0139
	Sine	Mean	0.8703	0.9504	0.8633	0.7399	0.8591	0.9448	0.8535	0.7204
		Variance	0.0018	0.0023	0.0017	0.0016	0.0107	0.0040	0.0152	0.0141
Musk	Gaussian	Mean	0.9239	0.5862	0.8804	0.6943	0.9180	0.5607	0.8604	0.6703
		Variance	0.0044	0.0280	0.0265	0.0198	0.0094	0.0563	0.0411	0.0378
	Sigmoid	Mean	0.9275	0.6090	0.8869	0.7106	0.9310	0.6201	0.8974	0.7220
		Variance	0.0083	0.0502	0.0316	0.0316	0.0102	0.0628	0.0416	0.0472
	Hardlim	Mean	0.9258	0.5810	0.9035	0.6995	0.9236	0.5659	0.9024	0.6893
		Variance	0.0029	0.0271	0.0127	0.0135	0.0029	0.0342	0.0225	0.0184
	Triangular	Mean	0.9212	0.5627	0.8839	0.6809	0.9168	0.5298	0.8859	0.6594
		Variance	0.0010	0.0129	0.0076	0.0073	0.0090	0.0519	0.0263	0.0292
	Sine	Mean	0.9231	0.6765	0.7949	0.7109	0.9247	0.6959	0.7910	0.7194
		Variance	0.0031	0.0446	0.0148	0.0180	0.0047	0.0460	0.0249	0.0199
Adult	Gaussian	Mean	0.8428	0.5424	0.7346	0.6007	0.8367	0.5323	0.7161	0.5889
		Variance	0.0027	0.0126	0.0064	0.0078	0.0027	0.0109	0.0131	0.0068
	Sigmoid	Mean	0.8525	0.5873	0.7454	0.6284	0.8467	0.5708	0.7327	0.6150
		Variance	0.0010	0.0049	0.0046	0.0029	0.0052	0.0121	0.0165	0.0118
	Hardlim	Mean	0.8470	0.5574	0.7424	0.6115	0.8401	0.5441	0.7232	0.5974
		Variance	0.0022	0.0126	0.0036	0.0075	0.0038	0.0114	0.0263	0.0082
	Triangular	Mean	0.8360	0.5069	0.7287	0.5795	0.8290	0.4920	0.7083	0.5648
		Variance	0.0019	0.0126	0.0024	0.0072	0.0020	0.0110	0.0175	0.0074
	Sine	Mean	0.8449	0.5593	0.7326	0.6089	0.8376	0.5460	0.7120	0.5945
		Variance	0.0009	0.0048	0.0051	0.0017	0.0042	0.0147	0.0137	0.0097
Diabetes	Gaussian	Mean	0.8473	0.6908	0.8353	0.6990	0.7603	0.5591	0.6854	0.5710
		Variance	0.0142	0.0323	0.0243	0.0230	0.0142	0.0627	0.0546	0.0232
	Sigmoid	Mean	0.7852	0.5691	0.7431	0.5986	0.7671	0.5608	0.7084	0.5775
		Variance	0.0069	0.0151	0.0160	0.0094	0.0295	0.0704	0.0959	0.0389
	Hardlim	Mean	0.9020	0.8052	0.8984	0.7985	0.7554	0.5894	0.6638	0.5781
		Variance	0.0015	0.0109	0.0155	0.0039	0.0318	0.0532	0.0743	0.0358
	Triangular	Mean	0.8799	0.7574	0.8751	0.7568	0.7503	0.5679	0.6621	0.5722
		Variance	0.0044	0.0165	0.0111	0.0080	0.0551	0.0628	0.0622	0.0506
	Sine	Mean	0.8223	0.6377	0.8027	0.6560	0.7747	0.5745	0.7137	0.5910
		Variance	0.0091	0.0235	0.0169	0.0157	0.0301	0.0619	0.0510	0.0379

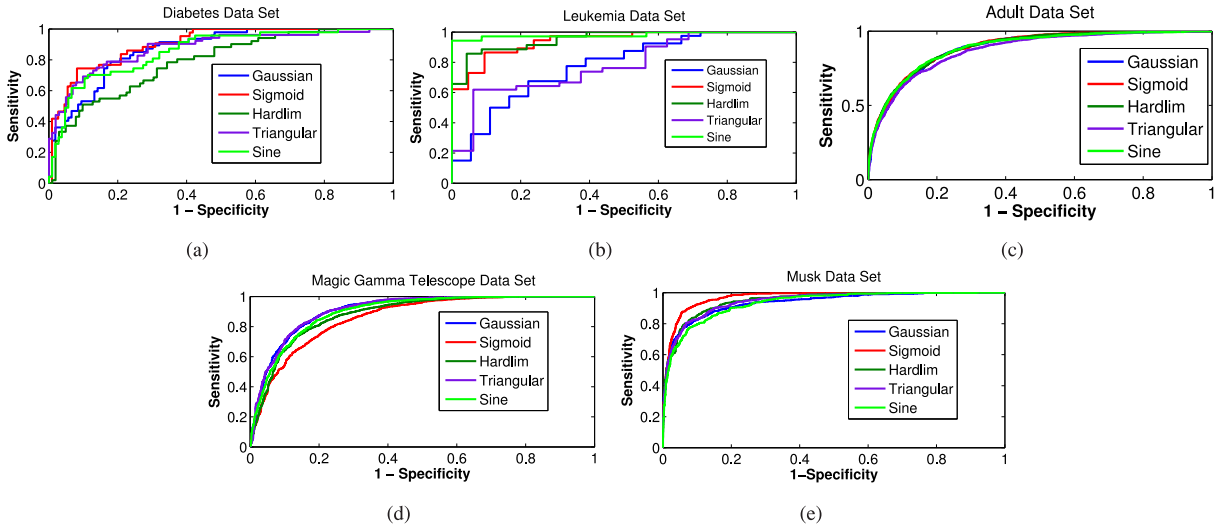


Fig. 2. ROC performance of the proposed algorithm with different activation functions. (a) Diabetes dataset. (b) Leukemia dataset. (c) Adult dataset. (d) MAGIC Gamma telescope dataset. (e) Musk dataset.

datasets and the proposed inverse-free ELM method is as efficient as the standard ELM in solving classification problems.

About the time efficiency of the proposed algorithm in running real problems, we have the following remark.

Remark 5: The proposed algorithm speeds up the computation by complementing standard ELM in determining the optimal hidden nodes. For example, for the situation that the original trial of standard ELM with N_1 hidden nodes but is

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VI
EXPERIMENTAL RESULTS OF THE PROPOSED ALGORITHM ON BIG DATASETS

		Gaussian		Sigmoid		Hardlim		Triangular		Sine	
		Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)	Accuracy	Time(s)
Scene	Rate(%)	72.30	28.8368	65.11	29.1409	69.18	29.4530	74.80	30.1784	74.53	29.0396
	Dev(%)	2.29	0.2978	1.10	0.1544	2.76	0.1986	2.21	0.6729	1.02	0.0552
MINST	Rate(%)	96.81	16.2631	93.24	17.8543	94.32	14.5627	97.41	14.8201	96.20	15.7561
	Dev(%)	0.18	0.4302	0.23	0.2316	0.08	0.0772	0.10	0.0882	0.07	0.1986
Connect	Rate(%)	78.99	27.0896	82.84	26.7152	79.05	23.9696	77.10	23.8058	80.55	25.8416
	Dev(%)	0.17	2.7688	0.36	1.3568	0.33	0.2316	0.06	1.2575	0.95	1.0479
Caltech	Rate(%)	29.83	303.4999	29.84	315.0986	22.27	310.0910	28.76	313.8194	30.31	311.1284
	Dev(%)	0.50	12.1340	0.48	2.7246	0.05	5.1956	0.95	5.7471	0.41	10.7221

not good enough to fit the training data with desired accuracy. The second trial uses an ELM with more hidden nodes, which is $N_2 > N_1$, to fit the same data. Using standard ELM, we have to compute the output weight with a computational burden dependent on N_2. Instead, the proposed algorithm is able to complement standard ELM to reach such a job with a computational burden depending on $N_2 - N_1$, which is smaller that N_2, especially for the situation that $(N_2 - N_1) \ll N_2$.

E. Further Experiments on Big Data

In this part, we run the proposed algorithm on datasets with a large number of instances to show the its performance on big data. We consider the following datasets: MNIST [61], Connect-4 [62], Caltech101/256 [63], and Scene15 [64], and report the training time, parameter sensitivity, and testing error (mean value and deviation) with various kernel functions, including Gaussian, sigmoid, hardlim, triangular, and sine. As to the datasets, the MNIST dataset is a large set on handwritten digits that is commonly used for training various image processing systems. The dataset is also widely used for training and testing in the field of machine learning. It contains 60 000 training images and 10 000 testing images. Half of the training set and half of the test set were taken from the National Institute of Standards and Technologies (NISTs) training dataset, while the other half of the training set and the other half of the test set were taken from NISTs testing dataset. The Connect-4 dataset contains all legal 8-ply positions in the game of Connect-4 in which either player has won yet, and in which the next move is not forced. It includes a total number of 67 557 instances with 42 attributes. The dataset Caltech101/256 is a set of 256 object categories containing a total of 30 607 images. The dataset Scene15 is composed of 15 scene categories including office, kitchen, living room, bedroom, store, industrial, tall building, inside cite, street, highway, coast, open country, mountain, forest, and suburb. Images in the dataset are about 250 $\times$ 300 resolution, with 210 to 410 images per class. This dataset contains a wide range of outdoor and indoor scene environments. The experimental results are summarized in Table VI, from which we can observe that the proposed algorithm bears a similar testing accuracy to state-of-the-art algorithms with a very fast training speed.

We also run extensive simulations to test the sensitivity of the testing accuracy with respect to the Tikhonov regulation parameter on the datasets including Scene15, MNIST, and Connect-4. The results are drawn in the following figure.

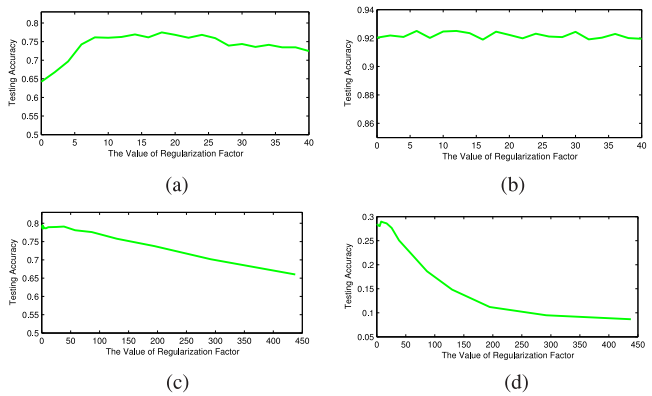


Fig. 3. Sensitivity of the testing accuracy with respect to the Tikhonov regulation parameter on different datasets. (a) Scene15 dataset. (b) MNIST dataset. (c) Connect-4 dataset. (d) Caltech256 dataset.

For the dataset Scene15, the average accuracy increases with the increase of the Tikhonov regulation parameter from 0 to 7, due to the increased generalization effect introduced by a larger regulation parameter. However, further increase of the Tikhonov regulation parameter does not receive additional performance improvement. Although Tikhonov regulation help increase the generalization ability of the model, but it is at the cost of optimality. Consequently as observed in Fig. 3, the average testing accuracy start to drop when further increase of the regulation parameter is applied from 20 to 40. For the Mixed National Institute of Standards and Technology dataset, we test the results using regulation parameters ranging from 0 to 40. The results show that the performance of the proposed algorithm is not sensitive to this parameter. For the Connect-4 dataset, with the increase of the regulation factor the testing accuracy first experiences an increase in a small range between 0 and 30 and then keeps dropping for regulation factors from 30 to 450. The results for the Caltech256 dataset also witness continuous decreases in testing accuracy after a short increase when the regulation parameter is tune from 0 through 450.

VI. CONCLUSION

In this paper, a novel algorithm removing the matrix inversion operation in the standard ELM but reaching exactly the same solution was proposed to solve regression and classification problems. The proposed algorithm gradually increases the number of nodes in the hidden layer of the ELM, progressively updates the connection weights in an optimal manner, and reaches the exactly the same solution in every step as the conventional ELM but without the computation

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

of matrix inversion. Time complexity of the proposed algorithm is analyzed theoretically. Both the monotonicity of the updating rule and the optimality of the obtained connection weights are proved theoretically. Numerical experiments show the effectiveness and accuracy of the proposed algorithm in solving real-world regression and classification problems.

REFERENCES

[1] S. Li, Y. Li, and Z. Wang, "A class of finite-time dual neural networks for solving quadratic programming problems and its winners-take-all application," *Neural Netw.*, vol. 39, pp. 27–39, Mar. 2013.

[2] Y. Li, S. Li, and Y. Ge, "A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments," *Neurocomputing*, vol. 104, pp. 170–179, Mar. 2013.

[3] S. Li, B. Liu, and Y. Li, "Selective positive feedback produces the winner-take-all competition in recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 301–309, Feb. 2013.

[4] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative-filtering for recommender systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[5] S. Li and Y. Li, "Nonlinearly activated neural network for solving time-varying complex Sylvester equation," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1397–1407, Aug. 2014.

[6] X. Luo, Y. Xia, and Q. Zhu, "Incremental collaborative filtering recommender based on regularized matrix factorization," *Knowl. Based Syst.*, vol. 27, pp. 271–280, Mar. 2012.

[7] J. Cao and Z. Lin, "Extreme learning machines on high dimensional and large data applications: A survey," *Math. Prob. Eng.*, Mar. 2015, Art. ID 103796. [Online]. Available: http://www.hindawi.com/journals/mpe/aa/103796/abs/

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Representations by Back-Propagating Errors*. Cambridge, MA, USA: MIT Press, 1988.

[9] P. J. Werbos, "*The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*," vol. 1. New York, NY, USA: Wiley, 1994.

[10] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[11] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[12] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2. Budapest, Hungary, 2004, pp. 985–990.

[13] L. L. C. Kasun, H. Zhou, G. B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 31–34, Dec. 2013.

[14] G. B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 274–281, Mar. 2003.

[15] B. Widrow, A. Greenblatt, Y. Kim, and D. Park, "The No-Prop algorithm: A new learning algorithm for multilayer neural networks," *Neural Netw.*, vol. 37, pp. 182–188, Jan. 2013.

[16] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[17] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[18] R. Zhang, Y. Lan, G. Huang, and Z. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, Feb. 2012.

[19] R. Zhang, Y. Lan, G. Huang, Z. Xu, and Y. C. Soh, "Dynamic extreme learning machine and its approximation capability," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2054–2065, Dec. 2013.

[20] G. Feng, G. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009.

[21] G. Feng, Y. Lan, X. Zhang, and Z. Qian, "Dynamic adjustment of hidden node parameters for extreme learning machine," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 279–288, Feb. 2015.

[22] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[24] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: An extreme learning approach," in *Proc. Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, 2010, pp. 315–320.

[25] Q. Liu, Q. He, and Z. Shi, "Extreme support vector machine classifier," in *Advances in Knowledge Discovery and Data Mining* (LNCS 5012), T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, Eds. Berlin, Germany: Springer, 2008, pp. 222–233.

[26] B. Frenay and M. Verleysen, "Parameter-insensitive kernel in extreme learning for non-linear support vector regression," *Neurocomputing*, vol. 74, no. 16, pp. 2526–2531, 2011.

[27] E. Parviainen, J. Riihimäki, Y. Miche, and A. Lendasse, "Interpreting extreme learning machine as an approximation to an infinite neural network," in *Proc. Int. Conf. Knowl. Disc. Inf. Retriev. (KDIR)*, Valencia, Spain, Oct. 2010, pp. 65–73.

[28] E. Cambria, G. B. Huang, L. L. C. Kasun, and H. Zhou, "Extreme learning machines: Trends controversies," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 30–59, Nov. 2013.

[29] Y. Miche, B. Schrauwen, and A. Lendasse, "Machine learning techniques based on random projections," in *Eur. Symp. Artif. Neural Netw.*, Bruges, Belgium, 2010, pp. 295–302.

[30] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, San Francisco, CA, USA, 2001, pp. 245–250.

[31] S. Dasgupta, "Experiments with random projection," in *Proc. 16th Conf. Uncertain. Artif. Intell.*, San Francisco, CA, USA, 2000, pp. 143–151.

[32] S. S. Vempala, *The Random Projection Method*, vol. 65. Providence, RI, USA: Amer. Math. Soc., 2004.

[33] P. Gastalo, R. Zunino, E. Cambria, and S. Decherchi, "Combining ELM with random projections," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 18–20, Apr. 2013.

[34] E. Goodman and D. Ventura, "Spatiotemporal pattern recognition via liquid state machines," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, 2006, pp. 3848–3853.

[35] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 131–144, Jan. 2011.

[36] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[37] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, no. 4, pp. 439–458, 2001.

[38] F. L. Lewis, A. Yesildirak, and S. Jagannathan, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Bristol, PA, USA: Taylor & Francis, 1998.

[39] Y. Zhang, W. Li, C. Yi, and K. Chen, "A weights-directly-determined simple neural network for nonlinear system identification," in *Proc. IEEE World Congr. Comput. Intell.*, Hong Kong, 2008, pp. 455–460.

[40] Y. Zhang and G. Ruan, "Bernoulli neural network with weights directly determined and with the number of hidden-layer neurons automatically determined," in *Proc. 6th Int. Symp. Neural Netw. Adv. Neural Netw.*, Wuhan, China, 2009, pp. 36–45.

[41] Q. Y. Zhu, A. K. Qin, P. N. Suganthan, and G. B. Huang, "Evolutionary extreme learning machine," *Pattern Recognit.*, vol. 38, no. 10, pp. 1759–1763, 2005.

[42] Z. Bai, G. B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1858–1870, Oct. 2014.

[43] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.

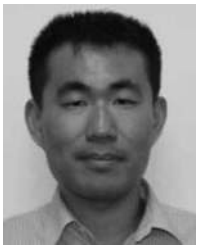
[44] A. Riccardi, F. Fernandez-Navarro, and S. Carloni, "Cost-sensitive AdaBoost algorithm for ordinal regression based on extreme learning machine," *IEEE Trans. Cybern.*, vol. 44, no. 10, pp. 1898–1909, Oct. 2014.

[45] J. Cao, T. Chen, and J. Fan, "Landmark recognition with compact BoW histogram and ensemble ELM," *Multimedia Tools Appl.*, vol. 9, no. 3, pp. 1–15, Jan. 2015.

[46] J. Cao, T. Chen, and J. Fan, "Fast Online learning algorithm for landmark recognition based on BoW framework," in *Proc. IEEE 9th Conf. Ind. Electron. Appl.*, Hangzhou, China, 2014, pp. 1163–1168.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[47] G. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cogn. Comput.*, vol. 6, no. 3, pp. 376–390, 2014.
[48] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.
[49] Y. Miche *et al.*, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.
[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[51] M. Lichman, *UCI Machine Learning Repository*, School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[52] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, May 2011.
[53] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy Build.*, vol. 49, pp. 560–567, Jun. 2012.
[54] R. Setiono and H. Liu, "A connectionist approach to generating oblique decision trees," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 3, pp. 440–444, Jun. 1999.
[55] M. A. Little, P. E. McSharry, S. J. Roberts, D. A. E. Costello, and I. M. Moroz, "Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection," *Biomed. Eng. Online*, vol. 6, no. 23, 2007.
[56] R. López-González, "Neural networks for variational problems in engineering," Ph.D. dissertation, Dept. Comput. Lang. Syst., Tech. Univ. Catalonia, Barcelona, Spain, Sep. 2008.
[57] T. G. Dietterich, R. H. Lathrop, and T. L. Perez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, nos. 1–2, pp. 31–71, 1997.
[58] R. K. Bock *et al.*, "Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope," *Nucl. Instrum. Methods Phys. Res. A*, vol. 516, pp. 511–528, Jan. 2004.
[59] T. R. Golub *et al.*, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, no. 5439, pp. 531–537, 1999.
[60] R. Kohavi, "Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid," in *Proc. 2nd Int. Conf. Knowl. Disc. Data Min.*, Portland, OR, USA, 1996, pp. 202–207.
[61] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[62] A. N. Burton and P. H. J. Kelly, "Performance prediction of paging workloads using lightweight tracing," *Future Gener. Comput. Syst.*, vol. 22, no. 7, pp. 784–793, 2006.
[63] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Comput. Vis. Image Understand.*, vol. 106, no. 1, pp. 59–70, 2007.
[64] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. New York, NY, USA, 2006, pp. 2169–2178.



Shuai Li (M'14) received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, Hefei, China, in 2005, the M.E. degree in automatic control engineering from the University of Science and Technology of China, Hefei, in 2008, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2014.

He joined Hong Kong Polytechnic University, Hong Kong, as a Research Assistant Professor. His current research interests include distributed estimation, optimization and control of networked systems, kinematic and dynamic analysis of robotic systems, and neural networks.

Dr. Li is currently on the Editorial Board of *Neural Computing and Applications*, and the *International Journal of Distributed Sensor Networks*.

Zhu-Hong You (M'14) received the B.E. degree in electronic information science and engineering from Hunan Normal University, Changsha, China, in 2005 and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2010.

From 2008 to 2009, he was a Visiting Research Fellow with the Center of Biotechnology and Information, Cornell University, Ithaca, NY, USA. He is currently a Post-Doctoral Fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His current research interests include neural networks, intelligent information processing, sparse representation, and its applications in bioinformatics.

Hongliang Guo received the Bachelor of Engineering degree in dynamic engineering and the Master of Engineering degree in dynamic control from the Beijing Institute of Technology, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the Stevens Institute of Technology, Hoboken, NJ, USA.

In 2011, he joined Almende, Rotterdam, The Netherlands, as a Post-Doctoral Researcher. In 2013, he joined Nanyang Technological University, as a Research Fellow. His current research interests include self-organizing systems and agent-based technologies.

Xin Luo (M'14) received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree from Beihang University, Beijing, China, in 2011, both in computer science.

He joined the College of Computer Science, Chongqing University, Chongqing, China, in 2011, where he is currently an Associate Professor of Computer Science and Engineering. His current research interests include recommender systems, social-network analysis, services computing, and bioinformatics.

Zhong-Qiu Zhao received the master's degree from the Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, China, in 2004, and the Ph.D. degree from the University of Science and Technology of China, Hefei, in 2007, both in pattern recognition and intelligent system.

From 2008 to 2009, he held a post-doctoral position in image processing with CNRS UMR6168 Lab Sciences de l'Information et des Systèmes, Paris, France. From 2013 to 2014, he was a Research Fellow in image processing with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. He is currently an Associate Professor with the Laboratory of Data Mining and Intelligent Computing, Hefei University of Technology, Hefei. His current research interests include pattern recognition, image processing, and computer vision.