Inverse-Free Incremental Learning Algorithms With Reduced Complexity for Regularized Extreme Learning Machine

HUFEI ZHU\(^1,2\) AND YANPENG WU\(^1\)

\(^1\)Department of Information Science and Engineering, Hunan First Normal University, Changsha 410205, China
\(^2\)College of Computer Science and Software, Shenzhen University, Shenzhen 518060, China

Corresponding author: Yanpeng Wu (xjzxwyp@hnfnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61621062, Grant 61773407, and Grant 61872408; in part by the Natural Science Foundation of Hunan Province of China under Grant 2019JJ40050; and in part by the Key Scientific Research Project of Education Department of Hunan Province of China under Grant 19A099.

ABSTRACT The existing inverse-free incremental learning algorithm for the regularized extreme learning machine (ELM) was based on an inverse-free algorithm to update the regularized pseudo-inverse, which was deduced from an inverse-free recursive algorithm to update the inverse of a Hermitian matrix. Before that recursive algorithm was applied in the existing inverse-free ELM, its improved version had been utilized in previous literatures. Then from the improved recursive algorithm to update the inverse, we deduce a more efficient inverse-free algorithm to update the regularized pseudo-inverse, from which we propose the inverse-free incremental ELM algorithm based on regularized pseudo-inverse. Usually the above-mentioned inverse is smaller than the pseudo-inverse, while in the processor units with limited precision, the recursive algorithm to update the inverse may introduce numerical instabilities. Then to further reduce the computational complexity, we also propose the inverse-free incremental ELM algorithm based on the \(LDL^T\) factors of the inverse, where the \(LDL^T\) factors are updated iteratively by the inverse \(LDL^T\) factorization. With respect to the existing inverse-free ELM, the proposed ELM based on regularized pseudo-inverse and that based on \(LDL^T\) factors are expected to require only \(\frac{3}{8+M}\) and \(\frac{1}{8+M}\) of complexities, respectively, where \(M\) is the output node number. The numerical experiments show that both the proposed ELM algorithms significantly accelerate the existing inverse-free ELM, and the speedup in training time is not less than 1.41. On the Modified National Institute of Standards and Technology (MNIST) Dataset, usually the proposed algorithm based on \(LDL^T\) factors is much faster than that based on regularized pseudo-inverse. On the other hand, in the numerical experiments, the original ELM, the existing inverse-free ELM and the proposed two ELM algorithms achieve the same performance in regression and classification, and result in the same solutions, which include the output weights and the output sequence for the same input sequence.

INDEX TERMS Extreme learning machine (ELM), recursive algorithms, inverse-free, inverse \(LDL^T\) factorization, regularized pseudo-inverse, incremental learning.

I. INTRODUCTION The extreme learning machine (ELM) [1] is an effective solution for single-hidden-layer feedforward networks (SLFNs) due to its unique characteristics, i.e., extremely fast learning speed, good generalization performance, and universal approximation capability [2]. Thus ELM has been widely applied in classification and regression [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Zia Uddin.

The incremental learning algorithm for ELM proposed in [2] achieves the universal approximation capability by adding hidden nodes one by one. However, it only updates the output weights for the newly added hidden node, and freezes the output weights of the existing hidden nodes. Accordingly, those output weights are no longer the optimal least-squares solution of the standard ELM algorithm. Then the inverse-free incremental ELM algorithm based on regularized pseudo-inverse was proposed in [4] to update the output weights of the added node and the existing nodes.
simultaneously, and the updated weights are identical to the optimal solution of the standard ELM algorithm. The ELM algorithm proposed in [4] was based on an inverse-free algorithm to update the regularized pseudo-inverse of the hidden layer output matrix. On the other hand, the incremental ELM algorithm based on generalized inverse (i.e., pseudo-inverse) was proposed in [5], which can only update the pseudo-inverse of the hidden layer output matrix recursively, and cannot be applied in the regularized ELM.

Recently, the incremental ELM has attracted more and more attention among researchers [6]–[10]. For food image classification, an adaptive reduced class incremental kernel ELM (ARCIKELM) was proposed in [6]. For multi-label learning with emerging new labels, an incremental kernel ELM was presented in [7], which can achieve the superior performance. For the ELM suffering from the coexistence of node fault and node noise, two fault tolerant incremental ELM algorithms were developed in [8] for the existence of node fault and node noise, two fault tolerant incremental ELM algorithms were developed in [8] for the regression problem, which are node fault tolerant incremental ELM (NFTCI-ELM) and node fault tolerant convex incremental ELM (NFTCI-ELM). Moreover, for multilayer ELM (MELM), the Multilayer Incremental Hybrid Cost Sensitive ELM with Multiple Hidden Output Matrix and Subnetwork Hidden Nodes was proposed in [9] to achieve better performance, and an incremental MELM with random search enhancement was proposed in [10], which selects the optimal node from multiple randomly generated nodes to possess a more compact network structure.

In [4], the inverse-free algorithm to update the regularized pseudo-inverse was deduced from an inverse-free recursive algorithm to update the inverse of a Hermitian matrix. Before the recursive algorithm to update the inverse was utilized in [4], it had been mentioned in previous literatures [11]–[15], while its improved version had been utilized in [15], [16]. Then in this article, from the improved recursive algorithm [15], [16] to update the inverse, we deduce a more efficient inverse-free algorithm to update the regularized pseudo-inverse, from which we propose the inverse-free incremental ELM algorithm based on regularized pseudo-inverse. Usually the above-mentioned inverse is smaller than the pseudo-inverse, while in the processor units with limited precision, the recursive algorithm to update the inverse may introduce numerical instabilities after a very large number of iterations [17]. Then to further reduce the computational complexity, we also propose the inverse-free incremental ELM algorithm based on the LDL$^T$ factors of the inverse, where the LDL$^T$ factors are updated iteratively by the inverse LDL$^T$ factorization proposed in [18].

This article is organized as follows. Section II describes the ELM model. In Section III, we introduce the existing inverse-free incremental ELM algorithm [4]. Then Section IV and Section V propose the inverse-free incremental ELM algorithm based on regularized pseudo-inverse and that based on LDL$^T$ factors, respectively. In Section VI, we compare the expected computational complexities of the existing and proposed inverse-free incremental ELM algorithms, and evaluate them by numerical experiments. Finally, we make conclusion in Section VII.

In the following sections, the superscripts $(\bullet)^T$ and $(\bullet)^{-1}$ denote the transpose and inverse operations of a matrix, respectively. Moreover, $\mathbf{0}_T$ is the $l \times 1$ zero column vector, while $I_l$ is the identity matrix of size $l$.

**II. ARCHITECTURE OF THE ELM**

In the ELM model, the $n$-th input node, the $i$-th hidden node, and the $m$-th output node can be denoted as $x_n$, $h_i$, and $z_m$, respectively, while all the $N$ input nodes, $l$ hidden nodes, and $M$ output nodes can be denoted as $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_N]^T \in \mathbb{R}^N$, $\mathbf{h} = [h_1 \ h_2 \ \cdots \ h_l]^T \in \mathbb{R}^l$, and $\mathbf{z} = [z_1 \ z_2 \ \cdots \ z_M]^T \in \mathbb{R}^M$, respectively. Accordingly the ELM model can be represented in a compact form as

$$\mathbf{h} = f(\mathbf{Ax} + \mathbf{d}) \quad (1)$$

and

$$\mathbf{z} = \mathbf{Wh}, \quad (2)$$

where the input weights $\mathbf{A} = [a_{ln}] \in \mathbb{R}^{l \times N}$, the biases of the hidden nodes $\mathbf{d} = [d_1 \ d_2 \ \cdots \ d_l]^T \in \mathbb{R}^l$, the output weights $\mathbf{W} = [w_{ml}] \in \mathbb{R}^{M \times l}$, and the activation function $f(\bullet)$ is entry-wise, i.e., $f(\mathbf{A}) = [f(a_{ln})] \in \mathbb{R}^{l \times N}$ for a matrix input $\mathbf{A} = [a_{ln}] \in \mathbb{R}^{l \times N}$. In (1), the activation function $f(\bullet)$ can be chosen as linear, sigmoid, Gaussian models, etc.

Assume there are totally $K$ distinct training samples, and let $\mathbf{x}_k \in \mathbb{R}^N$ and $\mathbf{z}_k \in \mathbb{R}^M$ denote the $k$-th training input and the corresponding $k$-th training output, respectively, where $k = 1, 2, \ldots, K$. Then the input sequence and the output sequence in the training set can be represented as

$$\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_K] \in \mathbb{R}^{N \times K}, \quad (3)$$

and

$$\mathbf{Z} = [\mathbf{z}_1 \ \mathbf{z}_2 \ \cdots \ \mathbf{z}_K] \in \mathbb{R}^{M \times K}, \quad (4)$$

respectively. We can substitute (3) into (1) to obtain the hidden layer output matrix

$$\mathbf{H} = f\left(\mathbf{AX} + \mathbf{1}^T \otimes \mathbf{d}\right), \quad (5)$$

where $\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \cdots \ \mathbf{h}_K] \in \mathbb{R}^{l \times K}$ is the value sequence of all $l$ hidden nodes, and $\otimes$ is the Kronecker product [4]. Then we can substitute (5) and (4) into (2) to obtain the actual training output sequence

$$\mathbf{Z} = \mathbf{WH}. \quad (6)$$

In ELM, only the output weight matrix $\mathbf{W}$ is adjustable, while $\mathbf{A}$ and $\mathbf{d}$ are randomly fixed. Denote the training output sequence (i.e., the desired output) as $\mathbf{Y}$. Then an ELM simply minimizes the estimation error

$$\mathbf{E} = \mathbf{Y} - \mathbf{Z} = \mathbf{Y} - \mathbf{WH} \quad (7)$$
by finding a least-squares solution \( W \) for the problem

\[
\min_{W} \| E \|_F^2 = \min_{W} \| Y - WH \|_F^2, \tag{8}
\]

where \( \| \cdot \|_F \) denotes the Frobenius norm.

For the problem (8), the unique minimum norm least-squares solution is \([1]\)

\[
W = YH^T \left( HH^T \right)^{-1}. \tag{9}
\]

To avoid over-fitting, the popular Tikhonov regularization [19], [20] can be utilized to modify (9) into

\[
W = YH^T \left( HH^T + k^2_0 I \right)^{-1}, \tag{10}
\]

where \( k^2_0 > 0 \) denotes the regularization factor. Obviously (9) is just the special case of (10) with \( k^2_0 = 0 \). Thus in what follows, we only consider (10) for the ELM with Tikhonov regularization.

### III. THE EXISTING INVERSE-FREE INCREMENTAL LEARNING ALGORITHM FOR THE REGULARIZED ELM

In machine learning, it is a common strategy to increase the hidden node number gradually until the desired accuracy is achieved. However, when this strategy is applied in ELM directly, the matrix inverse operation in (10) for the original ELM will be required when a few or only one extra hidden node is introduced, and then the algorithm will be computational prohibitive. Accordingly, an inverse-free strategy was proposed in [4], to update the output weights incrementally with the increase of the hidden nodes. In each step, the output weights obtained by the inverse-free algorithm are identical to the solution of the original ELM algorithm with the inverse operation.

Assume that in the ELM with \( l \) hidden nodes, we add one extra hidden node, i.e., the hidden node \( l + 1 \), which has the input weight row vector \( \tilde{a}_{l+1} = \left[ a_{l+1,1}, a_{l+1,2}, \ldots, a_{l+1,N} \right]^T \in \mathbb{R}^N \) and the bias \( \tilde{d}_{l+1} \). Then from (5), it can be seen that the extra row \( \tilde{h}_{l+1} = f \left( \tilde{a}_{l+1}X + \tilde{d}_{l+1} \right) \) needs to be added to \( H \), i.e.,

\[
H^{l+1} = \begin{bmatrix} H \mid \tilde{h}_{l+1} \end{bmatrix}, \tag{11}
\]

where \( H^i \) (\( i = l, l + 1 \)) denotes \( H \) for the ELM with \( i \) hidden nodes. In \( \tilde{a}_{l+1}, \tilde{h}_{l+1}, \tilde{d}_{l+1} \) and what follows, we add the overline to emphasize the extra vector or scalar, which is added to the matrix or vector for the ELM with \( l \) hidden nodes.

After \( H \) is updated by (11), the original ELM updates the output weights by (10) that involves an inverse operation. To avoid that inverse operation, the incremental learning algorithm for ELM in [4] utilizes an inverse-free algorithm to update the regularized pseudo-inverse

\[
B = H^T \left( HH^T + k^2_0 I \right)^{-1}, \tag{12}
\]

and then substitutes (12) into (10) to compute the output weights by

\[
W = YB. \tag{13}
\]

In [4], \( B^{l+1} \) (i.e., \( B \) for the ELM with \( l + 1 \) hidden nodes) is computed from \( B^l \) iteratively by

\[
B^{l+1} = \left[ B^l \mid \tilde{b}_{l+1} \right], \tag{14}
\]

where

\[
\tilde{b}_{l+1} = \frac{\left( \tilde{h}_{l+1}^THh_{l+1} + k^2_0 I - \tilde{h}_{l+1}B^lHh_{l+1} \right) B^l}{\tilde{h}_{l+1}B^lHh_{l+1} + k^2_0 I} \tag{15}
\]

and \( B^l \), the \((l + 1)^{th} \) column of \( B^{l+1} \), is computed by

\[
\tilde{b}_{l+1} = \frac{-B^lH\tilde{h}_{l+1} + \tilde{h}_{l+1}}{\tilde{h}_{l+1}B^lH\tilde{h}_{l+1} + k^2_0 I}. \tag{16}
\]

Let

\[
R = HH^T + k^2_0 I \tag{17}
\]

and

\[
Q = R^{-1} = \left( HH^T + k^2_0 I \right)^{-1}. \tag{18}
\]

Then we can write (12) as

\[
B = H^TQ. \tag{19}
\]

From (17) we have \( R^{l+1} = H^{l+1}(H^{l+1})^T + k^2_0 I^{l+1} \), into which we substitute (11) to obtain

\[
R^{l+1} = \begin{bmatrix} R^l & p_l^T \\ p_l & h_{l+1}^TH_{l+1} + k^2_0 \end{bmatrix}, \tag{20}
\]

where \( p_l \), a column vector with \( l \) entries, satisfies

\[
p_l = H\tilde{h}_{l+1}. \tag{21}
\]

The inverse-free recursive algorithm computes \( Q^{l+1} = (R^{l+1})^{-1} \) by equations (16), (13), (14) and (11) in [4], which can be written as

\[
\begin{align*}
\hat{Q}_l &= Q_l + \frac{Q_l p_l p^T_l Q_l}{h_{l+1}^TH_{l+1} + k^2_0 - p^T_l Q_l p_l}, \tag{22a} \\
\hat{t}_l &= -\frac{Q_l p_l}{h_{l+1}^TH_{l+1} + k^2_0}, \tag{22b} \\
\tau_l &= \frac{p^T_l Q_l p_l}{(h_{l+1}^TH_{l+1} + k^2_0)^2} + \frac{1}{h_{l+1}^TH_{l+1} + k^2_0}. \tag{22c}
\end{align*}
\]

and

\[
Q^{l+1} = \begin{bmatrix} \hat{Q}_l & \hat{t}_l \\ \hat{t}_l^T & \tau_l \end{bmatrix}, \tag{23}
\]

respectively. Notice that in (22) and (23), \( t_l \) is a column vector with \( l \) entries, and \( \tau_l \) is a scalar.

The existing inverse-free incremental learning algorithm for ELM with Tikhonov regularization has been summarized in [4, Algorithm 2], which will be described in the following Algorithm 1.
Algorithm 1 The Existing Inverse-Free Incremental Learning Algorithm in [4] for the Increment of Hidden Nodes in Regularized ELM

Require:
- Desired approximation error \( \eta^* \), regularization factor \( k_0^2 \), the input sequence \( X \in \mathbb{R}^{N \times K} \), the training output sequence \( Y \in \mathbb{R}^{M \times K} \), the initial ELM model with \( l_0 \) hidden nodes (input weights \( A^{(0)} \in \mathbb{R}^{l_0 \times N} \), biases \( d^{(0)} \in \mathbb{R}^{l_0} \), regularized pseudo-inverse \( B^{(0)} \in \mathbb{R}^{K \times l_0} \), output weights \( W^{(0)} \in \mathbb{R}^{M \times K} \)).
- \( \eta = MSE(E) \) //initial mean square error

Ensure:
- An ELM with \( l \) hidden nodes \( (A, d, W) \) to reach the desired approximation error \( \eta^* \).

1: \( l = l_0 \), \( A = A^{(0)} \), \( d = d^{(0)} \), \( B = B^{(0)} \), \( W = W^{(0)} \)
2: \( H = f(AX + I^{T} \odot d) \)
3: \( E = Y - WH \) //initial training error
4: \( \eta = MSE(E) \) //initial mean square error
5: while \( \eta > \eta^* \) do
6: \( l = l + 1 \)
7: \( \tilde{a} \leftarrow \) a random vector in \( \mathbb{R}^N \)
8: \( \tilde{d} \leftarrow \) a random scalar
9: \( \tilde{h}^T = f(\tilde{a}^T X + \tilde{d} I^{T}) \) //the extra row added to \( H \)
10: \( \tilde{B} \leftarrow \frac{(\tilde{h}^T \tilde{h} + k_0^2) - \tilde{h}^T \tilde{h} \tilde{h}^T}{\tilde{h}^T \tilde{h} + k_0^2} \)
11: \( b \leftarrow -\frac{\tilde{B} \tilde{h}^T h + k_0}{\tilde{h}^T \tilde{h} + k_0} \)
12: \( B \leftarrow \begin{bmatrix} B & b \end{bmatrix} \) //update \( B \)
13: \( W \leftarrow \tilde{Y} \tilde{B} \) //update \( W \)
14: \( H \leftarrow \begin{bmatrix} H \ 
\tilde{h}^T \end{bmatrix} \) //update \( H \) by adding the extra row \( \tilde{h}^T \)
15: \( E = Y - WH \) //training error
16: \( \eta = MSE(E) \) //mean square error
17: end while

IV. THE PROPOSED INCREMENTAL ELM ALGORITHM BASED ON REGULARIZED PSEUDO-INVERSE

Actually the inverse-free recursive algorithm by (22) and (23) had been mentioned in previous literatures [11]–[15], before it was deduced in [4] by utilizing the Sherman-Morrison formula and the Schur complement. That inverse-free recursive algorithm can be regarded as the application of the block matrix inverse lemma [11, p.30], and was called the lemma for inversion of block-partitioned matrix [12, Ch. 14.12], [13, equation (16)]. To develop multiple-input multiple-output (MIMO) detectors, the inverse-free recursive algorithm was applied in [13], [14], and its improved version was utilized in [15], [16].

A. THE PROPOSED INCREMENTAL ELM UPDATING REGULARIZED PSEUDO-INVERSE

In the improved version [15, 16], equation (22) has been simplified into [15, equation (20)]

\[
\begin{align*}
\tau_l &= 1/\left(\hat{h}_{l+1}^T \hat{h}_{l+1} + k_0^2 - p_l^T Q_l^T p_l\right) \\
\tau_l' &= -\tau_l Q_l^T p_l \\
Q_l' &= Q_l' + (1/\tau_l) t_l t_l^T.
\end{align*}
\]

Then we can utilize (24) to simplify (16) and (15) into

\[
\begin{align*}
b_{l+1} &= \tau_l \left(\hat{h}_{l+1} - B_l^T p_l\right) \\
\end{align*}
\]

and

\[
\begin{align*}
\tilde{B}_{l+1} &= B_l^T - \hat{h}_{l+1}^T \tilde{B}_{l+1}^T B_l' \\
\end{align*}
\]

respectively, where \( \tau_l \) can be computed by

\[
\tau_l = 1/\left(\hat{h}_{l+1}^T \hat{h}_{l+1} + k_0^2 - \hat{h}_{l+1}^T B_l' p_l\right).
\]

Moreover, from (25) and (26) we can deduce an efficient algorithm to update the output weight matrix \( W \), i.e.,

\[
W_{l+1} = \begin{bmatrix} \hat{W}_l \ 
\hat{w}_{l+1} \end{bmatrix},
\]

where

\[
\begin{align*}
\hat{W}_l &= W_l - \hat{h}_{l+1} \tilde{B}_{l+1}^T B_l' \\
\hat{w}_{l+1} &= \hat{Y} b_{l+1}.
\end{align*}
\]

B. DERIVATION OF SEVERAL EQUATIONS

In this subsection, let us derive (25), (26), (27), (28) and (29) in the proposed incremental ELM based on regularized pseudo-inverse.

Firstly, let us substitute (11) and (18) into (12) to obtain

\[
B_l' = \begin{bmatrix} (H_l')^T \ 
\tilde{h}_{l+1} \end{bmatrix} Q_l'^{+1}.
\]

We can substitute (24c) into (23), which is then substituted into (32) to obtain

\[
B_l'^{+1} = \begin{bmatrix} (H_l')^T \ 
\tilde{h}_{l+1} \end{bmatrix} \left[ Q_l'^{+1} + (1/\tau_l) t_l t_l^T \right] t_l^T,
\]

i.e.,

\[
B_l'^{+1} = \begin{bmatrix} (H_l')^T \ 
Q_l'^{+1} t_l t_l^T + \hat{h}_{l+1} t_l^T \end{bmatrix}^T.
\]

To deduce (25), denote the second entry in the right side of (33) as

\[
\hat{b}_{l+1} = (H_l')^T t_l + \tau_l \hat{h}_{l+1}.
\]
Algorithm 2 The Proposed Incremental ELM Based on Regularized Pseudo-Inverse

Require:
Desired approximation error \( \eta^* \), regularization factor \( k_0^2 \), the input sequence \( X \in \mathbb{R}^{N \times K} \), the training output sequence \( Y \in \mathbb{R}^{L \times K} \), the initial ELM model with \( l_0 \) hidden nodes (input weights \( A_{0} \in \mathbb{R}^{N \times N} \), biases \( d_{0} \in \mathbb{R}^{1 \times N} \), regularized pseudo-inverse \( B_{0} \in \mathbb{R}^{K \times l_{0}} \), output weights \( W_{0} \in \mathbb{R}^{M \times l_{0}} \)).

Ensure:
An ELM with \( l \) hidden nodes (\( A, d, W \)) to reach the desired approximation error \( \eta^* \).

1: \( l \leftarrow l_0 \), \( A \leftarrow A_{0} \), \( d \leftarrow d_{0} \), \( B \leftarrow B_{0} \), \( W \leftarrow W_{0} \)
2: \( H = f (AX + 1^T \otimes d) \)
3: \( E = Y - WH \) //initial training error
4: \( \eta = \text{MSE}(E) \) //initial mean square error
5: while (\( \eta > \eta^* \)) do
6: \( l \leftarrow l + 1 \)
7: \( \hat{a} \leftarrow \text{a random vector in } \mathbb{R}^{N} \)
8: \( \bar{d} \leftarrow \text{a random scalar} \)
9: \( h^T = f (\hat{a}^T \bar{X} + \bar{d}1^T) \) //the extra row added to \( H \)
10: \( \bar{b} \leftarrow \frac{h - BH \bar{h}}{h^T BH} \)
11: \( \bar{w} = Y \bar{b} \)
12: \( W \leftarrow [W - \bar{w}h^T B \bar{w}] \) //update \( W \)
13: \( \bar{B} \leftarrow B - \bar{b}h^T \)
14: \( B = [\bar{B} \bar{b}] \) //update \( B \)
15: \( H \leftarrow [H \ h^T] \) //update \( H \) by adding the extra row \( h^T \)
16: \( E = Y - WH \) //training error
17: \( \eta = \text{MSE}(E) \) //mean square error
18: end while

into which substitute (24b) to obtain
\[
\tilde{b}_{i+1} = -\tau_l (H^T Q) p_l + \tau_l \bar{h}_{i+1},
\]
and then substitute (19) into (35).

To deduce (26), substitute (24b) into the first entry in the right side of (33), and denote it as \( B_i = (H^T Q - (H^T Q) p_l + \tau_l \bar{h}_{i+1}) p_l Q_l \), i.e.,
\[
B_i = (H^T Q - (H^T T_l + \tau_l \bar{h}_{i+1})) p_l Q_l.
\]

Then substitute (34) into (36) to obtain
\[
B_i = (H^T Q) p_l - \tilde{b}_{i+1} p_l Q_l,
\]
into which substitute (21) to obtain
\[
\bar{B}_i = (H^T Q - \tilde{b}_{i+1} p_l Q_l.
\]

Finally we need to substitute (19) into (38).

To deduce (27), substitute (21) into (24a) to obtain
\[
\tau_l = 1/(h_{i+1}^2 + h_{i+1} + k_0^2 - H_{i+1}^T Q p_l),
\]
which substitute (19).

Finally let us deduce (28) and (29). We can substitute (26) into (14), and then substitute (14) into (13), to deduce (28) where \( \tilde{w}_{i+1} \) satisfies (29b) and
\[
\tilde{W}^l = Y B^l = Y b_{i+1} + \bar{b}_{i+1} Q_i.
\]

Then we can substitute (29b) and (13) into (39) to deduce (29a).

V. THE PROPOSED INCREMENTAL ELM ALGORITHM BASED ON LDL^T FACTORS

The pseudo-inverse \( B \) defined by (19) is computed from the unique inverse \( Q \), while usually \( Q \) is smaller than \( B \). Then we update \( Q \) instead of \( B \), to further reduce the computational complexity. Moreover, instead of the unique inverse \( Q \), we update the \( LDL^T \) factors of \( Q \) to improve numerical stabilities. Finally in this section, we propose the incremental ELM based on \( LDL^T \) factors (of the unique inverse).

A. THE ELM ALGORITHM UPDATING THE UNIQUE INVERSE INSTEAD OF THE PSEUDO-INVERSE

Since \( H \) is \( l \times K \), the pseudo-inverse \( B \) defined by (12) is \( K \times l \), and the unique inverse \( Q \) defined by (18) is \( l \times l \). In SLFNs, usually there are more training samples than hidden nodes, i.e., \( K > l \). Accordingly, usually the \( l \times l \) matrix \( Q \) is smaller than the \( K \times l \) matrix \( B \). Then to further reduce the computational complexity, we can update \( Q \) instead of \( B \), and utilize \( Q \) to compute the output weights \( W \). We update \( Q \) by (21), (24) and (23), and update \( W \) by (28) where \( \bar{W}^l \) and \( \bar{w}_{i+1} \) are computed from the entries in the updated \( Q \).

To compute \( \bar{W}^l \), we substitute (19) into (29a) to obtain
\[
\bar{W}^l = W^l - \tilde{w}_{i+1} h_{i+1}^T Q_i^l,
\]
in which we substitute (21) to obtain
\[
\bar{W}^l = W^l - \bar{w}_{i+1} p_l^T Q_i^l.
\]

Finally, we substitute (24b) into (40) to obtain
\[
\bar{W}^l = W^l + (\bar{w}_{i+1} / \tau_l) p_l^T.
\]

On the other hand, to compute \( \bar{w}_{i+1} \), we substitute (25) into (29b) to obtain
\[
\bar{w}_{i+1} = \tau_l \left( Y h_{i+1} - Y B^l p_l \right),
\]
into which substitute (13) (i.e., \( W = Y B \)) to obtain
\[
\bar{w}_{i+1} = \tau_l \left( Y h_{i+1} - W^l p_l \right).
\]

Notice that in (41) and (43), \( \tau_l \) and \( \eta^* \) are a vector and a scalar in \( Q^l \), respectively, as shown in (23).

B. THE PROPOSED INCREMENTAL ELM UPDATING LDL^T FACTORS OF THE UNIQUE INVERSE

Since the processor units are limited in precision, the recursive algorithm utilized to update \( Q \) may introduce numerical instabilities, which occurs after a very large number of iterations [17]. Thus instead of \( Q \), we update the \( LDL^T \) factors of \( Q \), since usually the \( LDL^T \) factorization is numerically stable [21]. The \( LDL^T \) factors include the upper-triangular \( L \) and the diagonal \( D \), which satisfy
\[
LDL^T = Q = R^{-1}.
\]
From (44), we can deduce
\[
L^{-T}D^{-1}L^{-1} = R, \tag{45}
\]
where the lower-triangular \(L^{-T}\) and the diagonal \(D^{-1}\) are the conventional LDL\(^T\) factors [21] of \(R\).

It can be seen from (44) that \(L\) and \(D\) are the inverse LDL\(^T\) factors \([18]\) of \(R\). Then \(L\) and \(D\) can be updated iteratively by the inverse LDL\(^T\) factorization proposed in \([18]\), i.e.,
\[
\begin{align}
L^{t+1} &= \begin{bmatrix} L^t & \tilde{t}_t \\ 0_f & 1 \end{bmatrix}, \tag{46a} \\
D^{t+1} &= \begin{bmatrix} D^t & 0_f \\ 0_f & \tau_t \end{bmatrix}, \tag{46b}
\end{align}
\]
where
\[
\begin{align}
\tilde{t}_t &= -L^tD^t(L^t)^T p_t, \tag{47a} \\
\tau_t &= 1/\left(\tilde{h}^T_{t+1}h_{t+1} + k_0^2 - p_t^T L^tD^t(L^t)^T p_t\right). \tag{47b}
\end{align}
\]
We can substitute (44) into (47a) and (47b), respectively, to show that \(\tilde{t}_t\) in (47a) and \(\tau_t\) in (24b) satisfy
\[
\tilde{t}_t = t_t/\tau_t, \tag{48}
\]
and \(\tau_t\) in (47b) is equal to \(\tau_t\) in (24a). Then we can substitute (48) into (41) to obtain
\[
\tilde{W}^t = W^t + \tilde{w}_{t+1}t_t. \tag{49}
\]
After updating \(L\) and \(D\), we can compute the output weights \(W\) by (43), (49) and (28).

Substitute (21) into (43) and (47) to obtain
\[
\tilde{w}_{t+1} = \tau_t \left(\tilde{Y}h_{t+1} - W^tH^t\tilde{h}_{t+1}\right) \tag{50}
\]
and
\[
\begin{align}
\tilde{t}_t &= -L^tD^t(L^t)^T H^t\tilde{h}_{t+1}, \tag{51a} \\
\tau_t &= 1/\left(\tilde{h}^T_{t+1}(H^T)^tH^t\tilde{h}_{t+1} + k_0^2 - \tilde{h}^T(H^T)^d(L^T)^dH^t\tilde{h}_{t+1}\right), \tag{51b}
\end{align}
\]
respectively. Also substitute (49) into (28) to obtain
\[
W^{t+1} = \begin{bmatrix} W^t + \tilde{w}_{t+1}t_t^T & \tilde{w}_{t+1} \end{bmatrix}. \tag{52}
\]
Lastly, let us utilize (51), (46), (50) and (52), to summarize the proposed inverse-free incremental learning algorithm for the regularized ELM, in the following Algorithm 3.

Notice that in Algorithm 3, we can utilize the conventional LDL\(^T\) factorization [21] to compute \((L^0)^{-T}\) and \((D^0)^{-1}\) from \(R^0\) by (45), and then compute the initial \(L^0\) and \(D^0\) from \((L^0)^{-T}\) and \((D^0)^{-1}\), respectively. On the other hand, we can also utilize the inverse LDL\(^T\) factorization in \([18]\), to compute \(L^0\) and \(D^0\) from \(R^0\) directly.

**Algorithm 3 The Proposed Incremental ELM Based on LDL\(^T\) Factors**

**Require:**
Desired approximation error \(\eta^*\), regularization factor \(k^*_0\), the input sequence \(X \in \mathbb{R}^{N \times K}\), the training output sequence \(Y \in \mathbb{R}^{M \times K}\), the initial ELM model with \(l_0\) hidden nodes (input weights \(A^0 \in \mathbb{R}^{l_0 \times N}\), biases \(d^0 \in \mathbb{R}^{l_0}\), regularized pseudo-inverse \(B^0 \in \mathbb{R}^{K \times l_0}\), output weights \(W^0 \in \mathbb{R}^{l_0 \times K}\)).

**Ensure:**
An ELM with \(l\) hidden nodes \((A, d, W)\) to reach the desired approximation error \(\eta^*\).

1: \(l = l_0, A = A^0, d = d^0, L = L^0, D = D^0, W = W^0\)
2: \(H = f(AX + 1^T \otimes d)\)
3: \(E = Y - WH\) //initial training error
4: \(\eta = \text{MSE}(E)\) //initial mean square error
5: \(\text{while}\) \((\eta > \eta^*)\) \text{do}
6: \(l \leftarrow l + 1\)
7: \(\tilde{a} \leftarrow\) a random vector in \(\mathbb{R}^N\)
8: \(\tilde{d} \leftarrow\) a random scalar
9: \(\tilde{h}^T = (\tilde{a}^T X + \tilde{d}^T 1)\) //the extra row added to \(H\)
10: \(\tilde{t} = -LD(L^T)H\tilde{h}\)
11: \(\tau = 1/\left(\tilde{h}^T\tilde{h} + k_0^2 - \tilde{h}^T(H^T)^dLD(L^T)H\tilde{h}\right)\)
12: \(L \leftarrow \begin{bmatrix} L^t & \tilde{t}_t^T \\ 0_f & 1 \end{bmatrix}, D \leftarrow \begin{bmatrix} D^t & 0_f \\ 0_f & \tau_t \end{bmatrix}\) //update \(L\) and \(D\)
13: \(\tilde{w} = \tau (Y\tilde{h} - WH\tilde{h})\)
14: \(W \leftarrow W + \tilde{w}t^T \tilde{t}^T\) //update \(W\)
15: \(H \leftarrow \begin{bmatrix} H & \tilde{h}^T \tilde{h}^T \end{bmatrix}\) //update \(H\) by adding the extra row \(\tilde{h}^T\)
16: \(E = Y - WH\) //training error
17: \(\eta = \text{MSE}(E)\) //mean square error
18: \text{end while}

**VI. COMPLEXITY ANALYSIS AND NUMERICAL EXPERIMENTS**

In this section, we analyze the computational complexities of the presented inverse-free ELM algorithms. Then we carry out numerical experiments to show the performance of the presented inverse-free ELM algorithms, and compare their complexities. In the rest of this section, the proposed incremental ELM algorithm based on regularized pseudo-inverse (i.e., Algorithm 2) and that based on LDL\(^T\) factors (i.e., Algorithm 3) will be denoted as the proposed ELM algorithms 1 and 2, respectively, for the sake of simplicity.
H. Zhu, Y. Wu: Inverse-Free Incremental Learning Algorithms With Reduced Complexity

TABLE 2. Experimental results of the existing and proposed ELM schemes for regression problems.

| Dataset + Kernel | Node Number | Weight Error (Existing: Proposed) | Output Error (training: testing) (Existing: Proposed) | Testing MSE |
|------------------|-------------|-----------------------------------|-----------------------------------------------|-------------|
| Airfoil          | 3           | 6e-16:6e-16                       | 8e-15:1e-14                                     | 4.8e-2      |
| Gaussian         | 100         | 2e-11:1e-8                        | 5e-12:4e-9                                      | 1.1e-2      |
| Energy           | 500         | 2e-9:4e-6                         | 1e-10:3e-7                                      | 7.7e-3      |
| Sigmoid          | 3           | 2e-14:1e-14                       | 7e-14:4e-14                                     | 3.0e-2      |
| Housing          | 100         | 3e-11:4e-8                        | 5e-12:5e-9                                      | 5.0e-3      |
| Sine             | 500         | 2e-9:1e-6                         | 1e-10:6e-8                                      | 3.7e-3      |
| Protein          | 3           | 2e-15:8e-16                       | 5e-14:3e-14                                     | 8.6e-2      |
| Triangular       | 100         | 2e-11:2e-9                        | 2e-14:1e-14                                     | 7.3e-3      |
|                  | 500         | 2e-9:3e-6                         | 9e-10:1e-6                                      | 5.4e-3      |

A. COMPLEXITY ANALYSIS OF THE PRESENTED INVERSE-FREE ELM ALGORITHMS

In this subsection, we compare the expected flops (floating-point operations) of the existing ELM algorithm in [4] and the proposed ELM algorithms 1 and 2. Obviously, $l_1 l_2 l_3 \approx 2 l_1 l_2 - 1$ flops are required to multiply a $l_1 \times l_2$ matrix by a $l_2 \times l_3$ matrix, and $l_1 l_2$ flops are required to sum two matrices in size $l_1 \times l_2$ [4].

In Table 1, we compare the flops of the existing ELM algorithm [4] and the proposed ELM algorithms 1 and 2. As in [4], the flops of the existing ELM algorithm do not include the $0(lK)$ entries for simplicity, since usually the ELM has large $K$ (the number of training examples) and $l$ (the number of hidden nodes). The flops of the proposed ELM algorithms do not include the entries that are $0(lK)$ or $0(MK)$. Since usually $M/l \approx 0$, it can easily be seen from Table 1 that with respect to the existing ELM algorithm, the proposed ELM algorithms 1 and 2 only require about $\frac{3}{8+M}$ and $\frac{1}{8+M}$ of flops, respectively.

Notice that in the proposed ELM algorithm 1, $\bar{h}_t^p_+ \mathbf{B}^t$ computed in (27) can be utilized in (26) and (29a). The dominant computational load of the proposed ELM algorithm 1 comes from (21), (27), (25) and (29b), of which the flops are $2kI$, $2kI$, $2kI$ and $2kM$, respectively. Moreover, in the proposed ELM algorithm 2, the dominant computational load comes from (21) and (43), of which the flops are $2kI$ and $2kM$, respectively.

B. NUMERICAL EXPERIMENTS

We follow the simulations in [4], to compare the existing inverse-free ELM algorithm and the proposed inverse-free ELM algorithms 1 and 2 on MATLAB software platform under a Microsoft-Windows Server with 128 GB of RAM. We utilize a fivefold cross validation to partition the datasets into training and testing sets. To measure the performance, we employ the mean squared error (MSE) for regression problems, and employ four commonly used indices for classification problems, i.e., the prediction accuracy (ACC), the sensitivity (SN), the precision (PE) and the Matthews correlation coefficient (MCC). Moreover, the regularization factor is set to $k_0^2 = 0.1$ to avoid over-fitting.

For the regression problem, we consider energy efficiency dataset [22], housing dataset [23], airfoil self-noise dataset [24], and physicochemical properties of protein dataset [25]. As Fig. 1 in [4], Fig. 1 in this article shows MSEs for the original ELM algorithm (that computes the output weights by (10)), the existing inverse-free ELM algorithm, and the proposed inverse-free ELM algorithms 1 and 2. Moreover, as Table 4 in [4], Table 2 shows the regression performance, where different activation functions are chosen for different datasets, which include Gaussian, sigmoid, sine and triangular. In Table 2, the weight error and the output error are defined as $\|W_1 - W_2\|_F$ and $\|Z_1 - Z_2\|_F$, respectively, where $W_1$ and $Z_1$ are computed by an inverse-free ELM algorithm, and $W_2$ and $Z_2$ are computed by the original ELM algorithm. We set the initial hidden node number to 2, and utilize the existing and proposed inverse-free ELM algorithms to

TABLE 3. Speedups in training time of the proposed inverse-free ELM algorithms 1 and 2 (i.e., Alg. 1 and Alg. 2) over the existing inverse-free ELM algorithm.

| Dataset + Kernel | Nodes Number | Speedups |
|------------------|--------------|----------|
| Airfoil          | 100          | 2.43     |
| Gaussian         | 500          | 2.61     |
| Energy           | 100          | 2.30     |
| Sigmoid          | 500          | 2.51     |
| Housing          | 100          | 2.73     |
| Sine             | 500          | 2.77     |
| Protein          | 100          | 2.54     |
| Triangular       | 500          | 2.66     |

177324
TABLE 4. Experimental results of the presented ELM algorithms for classification problems.

| Dataset   | Kernel | Mean/Variance | Training | Testing |
|-----------|--------|---------------|----------|---------|
|           |        |               | ACC      | SN      | PE      | MCC     | ACC      | SN      | PE      | MCC     |
| MAGIC     | Gaussian | Mean      | 0.8645   | 0.9472  | 0.8584  | 0.6975  | 0.8618   | 0.9459  | 0.8561  | 0.6914  |
|           |        | Variance   | 0.0019   | 0.0018  | 0.0018  | 0.0045  | 0.0068   | 0.0058  | 0.0064  | 0.0153  |
|           | Sigmoid | Mean      | 0.8602   | 0.9468  | 0.8536  | 0.6877  | 0.8588   | 0.9458  | 0.8525  | 0.6844  |
|           |        | Variance   | 0.0019   | 0.0019  | 0.0018  | 0.0044  | 0.0065   | 0.0049  | 0.0063  | 0.0146  |
|           | Hardlim | Mean      | 0.8312   | 0.9277  | 0.8315  | 0.6202  | 0.8270   | 0.9249  | 0.8284  | 0.6104  |
|           |        | Variance   | 0.0038   | 0.0046  | 0.0045  | 0.0088  | 0.0069   | 0.0066  | 0.0083  | 0.0147  |
|           | Triangular | Mean   | 0.8592   | 0.9419  | 0.8555  | 0.6852  | 0.8561   | 0.9398  | 0.8532  | 0.6780  |
|           |        | Variance   | 0.0023   | 0.0025  | 0.0024  | 0.0052  | 0.0060   | 0.0051  | 0.0066  | 0.0131  |
|           | Sine    | Mean      | 0.8640   | 0.9487  | 0.8569  | 0.6966  | 0.8620   | 0.9475  | 0.8552  | 0.6919  |
|           |        | Variance   | 0.0017   | 0.0016  | 0.0016  | 0.0040  | 0.0068   | 0.0058  | 0.0061  | 0.0152  |

TABLE 5. Experimental results of the existing and proposed inverse-free ELM algorithms on MNIST dataset.

| Algorithm | Gaussian Accuracy | Speedups | Sigmoid Accuracy | Speedups | Hardlim Accuracy | Speedups | Triangular Accuracy | Speedups | Sine Accuracy | Speedups |
|-----------|-------------------|----------|------------------|----------|------------------|----------|---------------------|----------|--------------|----------|
| Existing Alg. | 96.54%            |          | 94.67%           |          | 93.83%           |          | 96.20%              |          | 95.85%       |          |
| Proposed Alg. 1 | 96.54%            | 3.41     | 94.67%           | 3.77     | 93.83%           | 3.92     | 96.20%              | 3.14     | 95.85%       | 3.06     |
| Proposed Alg. 2 | 96.54%            | 26.28    | 94.67%           | 31.04    | 93.83%           | 33.69    | 96.20%              | 28.20    | 95.85%       | 34.42    |

add the hidden nodes one by one till the hidden node number reaches 500. Table 2 includes the simulation results for the hidden node numbers 3, 100 and 500.

From Fig. 1 and the last column of Table 2, it can be seen that the proposed inverse-free ELM algorithms 1 and 2 always achieve the same accuracy as the existing inverse-free ELM
in [4] and the original ELM based on matrix inversion. On the other hand, Table 2 shows that the weight error and the output error are less than $10^{-13}$ after 1 iteration (i.e., the node number 3), less than $10^{-10}$ after 98 iterations (i.e., the node number 100), and not greater than $2 \times 10^{-9}$ after 498 iterations (i.e., the node number 500). Then we can conclude that the original ELM, the existing inverse-free ELM in [4], and the proposed inverse-free ELM algorithms 1 and 2 all result in the same solutions, which include the output weight matrix and the output sequence for the same input sequence.

The speedups in training time of the proposed inverse-free ELM algorithms 1 and 2 over the existing inverse-free ELM algorithm are shown in Table 3, where we add just one node to reach 100 and 500 nodes, respectively, and we do 1000 simulations to compute the average training time. The speedups are computed by $T_{existing}/T_{proposed}$, i.e., the ratio between the training time of the existing ELM algorithm and that of the proposed ELM algorithm (1 or 2). As observed from Table 3, both the proposed algorithms 1 and 2 significantly accelerate the existing inverse-free ELM algorithm.

For the classification problem, we consider MAGIC Gamma telescope dataset [26], musk dataset [27], adult dataset [28] and diabetes dataset [25]. For each dataset, five activation functions are simulated, i.e., Gaussian, sigmoid, Hardlim, triangular and sine. In the simulations, the original ELM, the existing inverse-free ELM, and the proposed inverse-free ELM algorithms 1 and 2 all achieve the same performance, as listed in Table 4.

Lastly, in Table 5 we simulate the existing and proposed inverse-free ELM algorithms on the Modified National Institute of Standards and Technology (MNIST) dataset [29] with 60000 training images and 10000 testing images, to show the performance on big data. To give the testing accuracy, we set the initial hidden node number to 2000, and utilize the existing and proposed ELM algorithms to add hidden nodes one by one till the hidden node number reaches 2200. To give the speedups of the proposed inverse-free
ELM algorithms 1 and 2 over the existing inverse-free ELM algorithm, we compare the training time to reach 2200 nodes by adding one node, and do 500 simulations to compute the average training time.

As observed from Table 5, the existing and proposed inverse-free ELM algorithms bear the same testing accuracy, and usually the proposed algorithm 2 is much faster than the proposed algorithm 1 on MNIST Dataset. Moreover, from Table 3 and Table 5, we can conclude that both the proposed algorithms 1 and 2 significantly accelerate the existing inverse-free ELM algorithm, and the speedup in training time of the proposed inverse-free ELM algorithm (1 or 2) over the existing inverse-free ELM algorithm is not less than 1.41.

## VII. CONCLUSION

In this article, we utilize the improved recursive algorithm [15], [16] that updates the inverse, to deduce an inverse-free algorithm to update the regularized pseudo-inverse, which is more efficient than the corresponding algorithm utilized in the existing inverse-free incremental learning algorithm [4] for the regularized ELM. Accordingly, we propose the inverse-free incremental ELM algorithm based on regularized pseudo-inverse, which reduces the computational complexity of the existing inverse-free incremental ELM algorithm in [4].

Usually the unique inverse is smaller than the pseudo-inverse, while for the processor units with limited precision, the recursive algorithm to update the unique inverse may introduce numerical instabilities after a very large number of iterations [17]. Then to further reduce the computational complexity, we update the LDL-T factors of the unique inverse instead of the pseudo-inverse, by the inverse LDL-T factorization proposed in [18]. Accordingly, we propose the inverse-free incremental ELM algorithm based on LDL-T factors (of the unique inverse).

With respect to the existing inverse-free ELM algorithm, the proposed ELM algorithm based on regularized pseudo-inverse and that based on LDL-T factors are expected to require only \( \frac{3}{2} M^2 \) and \( \frac{1}{3} M^3 \) of flops, respectively, where \( M \) is the output node number. The numerical experiments show that both the proposed ELM algorithms significantly accelerate the existing inverse-free ELM, and the speedup in training time is not less than 1.41. On MNIST Dataset, usually the proposed algorithm based on LDL-T factors is much faster than that based on regularized pseudo-inverse. On the other hand, in the numerical experiments, the original ELM, the existing inverse-free ELM and the proposed two ELM algorithms achieve the same performance in regression and classification, and result in the same solutions, which include the output weights and the output sequence for the same input sequence.

In this article, we focus on the incremental learning for the regularized ELM, and propose two efficient inverse-free algorithms to add hidden nodes. However, usually it is also required to prune redundant hidden nodes [20], [30]–[33] by decremental learning algorithms in machine learning. Accordingly, our future works will include an efficient algorithm to remove hidden nodes in the regularized ELM.

## REFERENCES

[1] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: Theory and applications,” Neurocomputing, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[2] 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.

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

[4] S. Li, Z.-H. You, H. Guo, X. Luo, and Z.-Q. Zhao, “Inverse-free extreme learning machine with optimal information updating,” IEEE Trans. Cybern., vol. 46, no. 5, pp. 1229–1241, May 2016.

[5] B. Jin, Z. Jing, and H. Zhao, “Incremental and decremental extreme learning machine based on generalized inverse,” IEEE Access, vol. 5, pp. 20852–20865, 2017.

[6] G. A. Tahir and C. K. Liu, “An open-ended continual learning for food recognition using class incremental extreme learning machines,” IEEE Access, vol. 8, pp. 82328–82346, 2020.

[7] Y. Kongosrot, P. Horata, and P. Musiakawan, “An incremental kernel extreme learning machine for multi-label learning with emerging new labels,” IEEE Access, vol. 8, pp. 46055–464070, 2020.

[8] H. C. Leung, C. S. Leung, and E. W. M. Wong, “Fault and noise tolerance in the incremental extreme learning machine,” IEEE Access, vol. 7, pp. 155171–155183, 2019.

[9] D. Wu, Z. Qu, F. Guo, and Q. Wan, “Multilayer incremental hybrid cost-sensitive extreme learning machine with multiple hidden output matrix and subnetwork hidden nodes,” IEEE Access, vol. 7, pp. 118422–118434, 2019.

[10] J. Liu, X. Liu, C. Liu, B. T. Le, and D. Xiao, “Random search enhancement of incremental regularized multiple hidden layers ELM,” IEEE Access, vol. 7, pp. 36866–36878, 2019.

[11] H. Lütkepohl, Handbook of Matrices. New York, NY, USA: Wiley, 1996.

[12] T. K. Mood and W. C. Stirling, Mathematical Methods and Algorithms for Signal Processing. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.

[13] L. Szczechinski and D. Massicotte, “Low complexity adaptation of MIMO MMSE receivers, implementation aspects,” in Proc. Global Commun. Conf. (Globecom), St. Louis, MO, USA, Nov. 2005, pp. 1–6.

[14] Y. Shang and X.-G. Xia, “On fast recursive algorithms for V-BLAST with optimal ordered SIC detection,” IEEE Trans. Wireless Commun., vol. 8, no. 6, pp. 2860–2865, Jun. 2009.

[15] H. Zhu, W. Chen, and F. She, “Improved fast recursive algorithms for V-BLAST and G-STBC with novel efficient matrix inversion,” in Proc. IEEE Int. Conf. Commun., Jun. 2009, pp. 211–215.

[16] H. Zhu, W. Chen, B. Li, and F. Gao, “A fast recursive algorithm for G-STBC,” IEEE Trans. Commun., vol. 59, no. 8, pp. 2084–2089, Aug. 2011.

[17] J. Benesty, Y. Huang, and J. Chen, “A fast recursive algorithm for optimum sequential signal detection in a BLAST system,” IEEE Trans. Signal Process., vol. 51, no. 7, pp. 1722–1730, Jul. 2003.

[18] H. Zhu, W. Chen, and B. Li, “Efficient square-root and division free algorithms for inverse LDLT factorization and the wide-sense givens rotation with application to V-BLAST,” in Proc. IEEE 72nd Veh. Technol. Conf. Fall, 2010, pp. 6–9.

[19] Y. Mische, 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, Sep. 2011.

[20] Y. Mische, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, “QP-ELM: Optimally pruned extreme learning machine,” IEEE Trans. Neural Netw., vol. 21, no. 1, pp. 158–162, Jan. 2010.

[21] G. H. Golub and C. F. Van Loan, Matrix Computations, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[22] A. Tnasaa, M. and A. Xifara, “Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools,” Energy Buildings, vol. 49, pp. 560–567, Jun. 2012.

[23] 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.

[24] R. L. Gonzalez, “Neural networks for variational problems in engineering,” Ph.D. dissertation, Dept. Comput. Lang. Syst., Tech. Univ. Catalonia, Barcelona, Spain, Sep. 2008.
[25] M. Lichman, “UCI mach. learning repository,” School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[26] R. K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, and W. Wittek, “Methods for multidimensional event classification: A case study using images from a Cherenkov gamma-ray telescope,” Nucl. Instrum. Methods Phys. Res. A, Accel. Spectrom. Detect. Assoc.Equip., vol. 516, nos. 2–3, pp. 511–528, Jan. 2004.

[27] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, “Solving the multiple instance problem with axis-parallel rectangles,” Artif. Intell., vol. 89, nos. 1–2, pp. 31–71, Jan. 1997.

[28] 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.

[29] 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.

[30] E. D. Karnin, “A simple procedure for pruning back-propagation trained neural networks,” IEEE Trans. Neural Netw., vol. 1, no. 2, pp. 239–242, Jun. 1990.

[31] G. Castellano, A. M. Fanelli, and M. Petillo, “An iterative pruning algorithm for feedforward neural networks,” IEEE Trans. Neural Netw., vol. 8, no. 3, pp. 519–531, May 1997.

[32] X. Zeng and D. S. Yeung, “Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure,” Neurocomputing, vol. 69, nos. 7–9, pp. 825–837, Mar. 2006.

[33] H.-J. Kong, Y.-S. Ong, A.-H. Tan, and Z. Zhu, “A fast pruned-extreme learning machine for classification problem,” Neurocomputing, vol. 72, nos. 1–3, pp. 359–366, Dec. 2008.

HUFEI ZHU received the B.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 1998, the M.E. degree in computing from the National University of Singapore, Singapore, in 2004, and the Ph.D. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2014. He is currently with the Department of Information Science and Engineering, Hunan First Normal University, Changsha, China. His research activity has led to more than 80 patents granted in China, USA, Europe, Russia, Japan, South Korea, India, etc., and also led to numerous publications in leading international journals and conference proceedings. His current research interests include neural networks, signal processing, and wireless communications.

YANPENG WU received the Ph.D. degree in energy engineering from Central South University, Changsha, China, in 2016. He is currently a Professor with the Department of Information Science and Engineering, Hunan First Normal University, Changsha. His current research interests include neural networks, image processing, and computer vision.

* * *