Neurocomputing 241 (2017) 1–17

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Gram–Schmidt process based incremental extreme learning machine

Yong-Ping Zhao[a,*], Zhi-Qiang Li[a], Peng-Peng Xi[a], Dong Liang[b], Liguo Sun[c], Ting-Hao Chen[d]

[a] Jiangsu Province Key Laboratory of Aerospace Power Systems, College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[b] College of Physics and Electromechanics, Xiamen University, Xiamen 361005, China
[c] School of Aeronautic Science and Engineering, Beihang University, Beijing 100091, China
[d] Guangdong Maritime Safety Administration, Guangzhou 510260, China

ARTICLE INFO

Article history:
Received 16 June 2016
Revised 23 December 2016
Accepted 18 January 2017
Available online 27 January 2017

Communicated by Wang Gang

Keywords:
Extreme learning machine
Incremental learning
QR decomposition
Gram–Schmidt process

ABSTRACT

To compact the architecture of extreme learning machine (ELM), two incremental learning algorithms are proposed in this paper. The previous incremental learning algorithms for ELM recruit hidden nodes randomly, which is equivalent to implementing a random selection from a candidate set of infinite size. Hence, it is impossible to recruit *good* hidden nodes, and thus it usually requires more hidden nodes than traditional neural networks to achieve matched performance. To improve the quality of the hidden nodes recruited, an incremental learning algorithm for ELM is presented based on Gram–Schmidt process (GSI-ELM), which recruits the *best* hidden node from a random subset of fixed size via defining an evaluating criterion at each learning step. However, the "nesting effect" exists in the GSI-ELM, that is to say, the hidden nodes once recruited by GSI-ELM can not be later discarded. To treat this "nesting problem", the improved GSI-ELM (IGSI-ELM) is generated with an elimination mechanism. At each learning step IGSI-ELM eliminates the *worst* hidden node from the already-recruited group if it is not the newly-recruited one. Finally, to verify the efficacy and feasibility of the proposed algorithms, i.e. GSI-ELM and IGSI-ELM, in this paper, experiments on regression and classification benchmark data sets are investigated.

1. Introduction

Extreme learning machine (ELM) [1,2] has been a popular and powerful tool for training single-hidden-layer feedforward networks (SLFNs) in the past few years. In ELM, input network weights and hidden biases are randomly generated while output weights are obtained as the result of a linear optimization problem which is usually calculated analytically using the Moore–Penrose generalized inverse. ELM proposes an alternative to the popular gradient descent-based algorithms like backpropagation [3], which are well-known for being slow. Compared to other machine learning algorithms, ELM shows advantages in computational cost, generalization performance, and implementation [2]. Hence, it has attracted a great deal of attention from the machine learning community and obtained a wide range of applications in classification and regression.

Since ELM was proposed, two topics are popular. One is to cope with the sequential learning issues using ELM. As known, ELM is originally designed for batch learning problems. However, for real-world applications where new data arrive sequentially, ELM has to gather old and new data together to retrain a model from scratch so as to incorporate the new information. This is a very time-consuming process and impairs the most obvious characteristic of ELM, i.e., extremely fast learning speed. To efficiently and effectively deal with problems with sequential data, the sequential ELMs were investigated [4–6], which are able to learn data one-by-one or chunk-by-chunk with fixed or varying chunk length. To implement sequential ELMs smoothly, some meta-parameters need to be initialized, e.g., the number of hidden nodes. Due to the fact that ELM generates hidden layer randomly, it usually needs more hidden nodes than traditional neural networks to achieve comparable performance. It is found that some of the hidden nodes in such networks may play a very minor role in the network output and thus may eventually increase the network complexity. More importantly, large network size leads to longer running time in the testing phase of ELM, which may hamper its efficient deployment in some testing time sensitive scenarios. Thus, another topic on improving the compactness of ELM has attracted great interest. To solve this problem, two different strategies are usually pursued. The first refers to constructive algorithms [7–14], which begin with a small initial network and gradually recruit new hidden nodes until some stopping criterions are met.

* Corresponding author.
E-mail addresses: y.p.zhao@163.com, y.p.zhao@nuaa.edu.cn (Y.-P. Zhao).

http://dx.doi.org/10.1016/j.neucom.2017.01.049
0925-2312/

In contrast, the second strategy is called destructive algorithms, also known as pruning algorithms [15–19], in which a network with a larger than necessary size is initially trained, and then the redundant or less effective hidden nodes are gradually removed until the performance required deteriorates.

Belonging to the first strategy aforementioned, the incremental learning plays an important role in improving the compactness of ELM. Huang et al. firstly proposed an incremental extreme learning machine (HI-ELM) [7], which outperforms many popular incremental learning algorithms such as resource-allocating network (RAN) [20] and minimal RAN [21,22]. HI-ELM randomly generates the hidden nodes and analytically calculates the output weights of ELM. However, HI-ELM does not recalculate the output weights of all the existing nodes when a new node is recruited. When Barron's convex optimization learning method [23] is incorporated into HI-ELM, a convex incremental ELM (CI-ELM) was presented. Different from HI-ELM, CI-ELM recalculates the output weights of the existing hidden nodes after a new one is added, and CI-ELM can achieve faster convergence rate and more compact network architecture than HI-ELM while retaining the HI-ELM's simplicity and efficiency. Based on HI-ELM, an enhanced method for incremental ELM (EI-ELM)[9] was developed, in which at each learning step several hidden nodes are randomly generated and among them the hidden node resulting in the largest residual error decreasing will be added to the existing network and the output weight of the network will be calculated in a simple way as in HI-ELM. Subsequently, an error minimized incremental ELM (EMI-ELM) [10] was proposed, which adds random hidden nodes one by one or group by group (with varying group size) and updates the output weights incrementally. Recently, a computationally competitive incremental algorithm for ELM based on QR decomposition was proposed (QRI-ELM) [12]. Compared to EMI-ELM, QRI-ELM accelerates the training speed using Gram–Schmidt process and keeps the same generalization performance. Following the spirit of QRI-ELM, this paper further makes two main contributions as follows:

(1) The Gram–Schmidt process based incremental ELM (GSI-ELM): QRI-ELM at each learning step randomly selects one hidden node. This process can be regarded as a hidden node recruited randomly from a candidate set of infinite size. Hence it is impossible to recruit the *best* hidden node at each learning step. In this paper GSI-ELM realizes the incremental learning based on the Gram–Schmidt process like QRI-ELM, but a probabilistic trick [24–26] is utilized, which considers a random subset of fixed size, say κ, and picks the best hidden node from this set according to some criterion at each learning step. There are two benefits of implementing this trick. One is that a *better* hidden node is recruited at each learning step instead of a random choice. Another is that the dilemma of recruiting the *best* hidden node from the candidate set of infinite size is sidestepped. In [24,25], they proved that this trick could suffice to obtain an estimate that is better than 95% of all other estimates with $1-0.05$ probability if $\kappa = 59 = \left\lceil \frac{\log(0.05)}{\log(0.95)} \right\rceil$.

(2) The improved GSI-ELM (IGSI-ELM): Evidently, GSI-ELM suffers from the so-called "nesting effect" [27]. It means that the hidden nodes once recruited by GSI-ELM cannot be later discarded. To treat this "nesting problem", IGSI-ELM is proposed. In IGSI-ELM, if one hidden node is recruited at one learning step, then all the existing hidden nodes will be reevaluated again according to some evaluation criterion and the *worst* hidden node is picked out. If the *worst* hidden node is not the newly-recruited one, it will be eliminated, which is equivalent to the *worst* hidden node replaced with the newly-recruited one. Otherwise, any hidden node is not eliminated. When this elimination mechanism occurs, the number of hidden nodes keeps constant while the performance improving, which is especially suitable for the testing time sensitive scenarios. It is also consistent with the Occam's razor "plurality must never be posited without necessity" [28]. As thus, the "nesting problem" is treated to some degree.

To investigate the effectiveness and feasibility of the proposed GSI-ELM and IGSI-ELM, experiments on benchmark data sets including regression and classification are done. By means of comprehensive comparison, we show that GSI-ELM and IGSI-ELM outperform the other incremental learning algorithms in terms of the compactness of ELM.

The remainder of this paper is organized as follows. In Section 2, the traditional ELM is briefly introduced and its solution using QR decomposition is given. Section 3 depicts the QRI-ELM algorithm in detail. To improve the quality of the hidden nodes recruited in QRI-ELM, an evaluating criterion is defined to recruit the *best* hidden node from a random subset of fixed size, thus yielding GSI-ELM in Section 4. In Section 5, to overcome the "nesting effect" existing in GSI-ELM, IGSI-ELM is proposed, in which an elimination mechanism is introduced by defining some evaluation criterion. To test the effectiveness of the proposed algorithms in this paper, experiments on regression and classification benchmark data sets are carried out in Section 6. Finally, conclusions follow.

2. ELM

Considering an SLFN with L hidden nodes and activation function $h(\cdot)$, its output of $\boldsymbol{x} \in \Re^n$ is governed by

$$f(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\theta}_i h(\boldsymbol{a}_i, b_i, \boldsymbol{x}) \tag{1}$$

where $\boldsymbol{a}_i \in \Re^n$ and b_i are the learning parameters of hidden nodes, $\boldsymbol{\theta}_i$ is the weight connecting the ith hidden node to the output nodes. In ELM, the input weights $\boldsymbol{a}_i$ and hidden biases b_i are randomly generated. Given a set of training data $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^N \in \Re^n \times \Re^m$, ELM lets the network outputs equal the targets, so the following compact formulation is got:

$$\boldsymbol{H}\boldsymbol{\Theta} = \boldsymbol{T} \tag{2}$$

where

$$\boldsymbol{H} = \begin{bmatrix} h(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & h(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & h(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_1, \cdots, \boldsymbol{h}_L \end{bmatrix} \tag{3}$$

$$\boldsymbol{\Theta} = \left[\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L\right]^\top \tag{4}$$

and

$$\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_N]^\top \tag{5}$$

Here, $\boldsymbol{H}$ is the so-called hidden nodes output matrix. Finding the solution of Eq. (5) in a least square sense is equivalent to solving the following optimal problem

$$\min_{\boldsymbol{\Theta}} \left\{ J = \|\boldsymbol{H}\boldsymbol{\Theta} - \boldsymbol{T}\|_F^2 \right\} \tag{6}$$

The minimal norm least square solution of (6) is

$$\hat{\boldsymbol{\Theta}} = \boldsymbol{H}^\dagger \boldsymbol{T} \tag{7}$$

where $\boldsymbol{H}^\dagger$ is the Moore–Penrose generalized inverse of matrix $\boldsymbol{H}$. Different methods can be used to calculate Moore–Penrose generalized inverse of a matrix [29]: orthogonal projection method, iterative method, and QR decomposition. The orthogonal projection method is used in two cases: i) $\boldsymbol{H}^\dagger = (\boldsymbol{H}^\top\boldsymbol{H})^{-1}\boldsymbol{H}^\top$ when $\boldsymbol{H}^\top\boldsymbol{H}$ is nonsingular; ii) $\boldsymbol{H}^\dagger = \boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{H}^\top)^{-1}$ when $\boldsymbol{H}\boldsymbol{H}^\top$ is nonsingular. Generally, in real applications $L < N$, so the first case is commonly used. For the orthogonal projection method, the key step of finding $\boldsymbol{H}^\dagger$ is to calculate $(\boldsymbol{H}^\top\boldsymbol{H})^{-1}$ or $(\boldsymbol{H}\boldsymbol{H}^\top)^{-1}$. From the numerically computational viewpoint, the reliability and stability of computing $(\boldsymbol{H}^\top\boldsymbol{H})^{-1}$ or $(\boldsymbol{H}\boldsymbol{H}^\top)^{-1}$ is closely related to its condition number, defined as

$$\tau\left(\boldsymbol{H}\boldsymbol{H}^\top\right) = \tau\left(\boldsymbol{H}^\top\boldsymbol{H}\right) = \tau^2(\boldsymbol{H}) = \frac{\mu_{\max}^2}{\mu_{\min}^2} \tag{8}$$

where $\tau(\cdot)$ represents the condition number of a matrix, $\mu_{\max}$ and $\mu_{\min}$ represent the maximum and minimum nonzero singular values of $\boldsymbol{H}$, respectively. Generally, the larger the condition number is, the less reliable the numerical result becomes. If the condition number of $\boldsymbol{H}$ is large, say, $\tau(\boldsymbol{H}) = 10^{10}$, then $\boldsymbol{H}^\top\boldsymbol{H}$ or $\boldsymbol{H}^\top\boldsymbol{H}$ will become ill-conditioned. In this situation, the orthogonal projection method may give a numerically inaccurate solution and even generate a wrong one. Hence, it is wise to sidestep the large condition number as much as possible during the process of calculating the matrix inverse. QR decomposition is a good choice of calculating $\boldsymbol{H}^\dagger$, given as

$$\boldsymbol{H}^\dagger = \boldsymbol{R}^{-1}\boldsymbol{Q}^\top \tag{9}$$

where

$$\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{H} \tag{10}$$

Here, $\boldsymbol{Q}$ is a matrix with orthogonal columns satisfying $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}$, $\boldsymbol{R}$ is an upper triangular matrix. From (10), we know $\tau(\boldsymbol{R}) = \tau(\boldsymbol{H})$. If equation (10) has already been computed, then

$$\hat{\boldsymbol{\Theta}} = \boldsymbol{R}^{-1}\boldsymbol{Q}^\top\boldsymbol{T} \tag{11}$$

There are several methods for QR decomposition [29], such as Gram–Schmidt process, Householder transformation, or Givens rotations. For batch learning problems, the aforementioned three methods are all competent. More importantly, Gram–Schmidt process shows advantage in solving incremental learning problems, and therefore QRI-ELM was developed [12].

3. QRI-ELM

In QRI-ELM, the hidden nodes are incrementally recruited one by one. Assume that k hidden nodes have already been recruited, then Eq. (10) becomes $\boldsymbol{Q}_k\boldsymbol{R}_k = \boldsymbol{H}_k$, where $\boldsymbol{H}_k = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$, $\boldsymbol{Q}_k = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_k]$, and

$$\boldsymbol{R}_k = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ & r_{22} & \cdots & r_{2k} \\ & & \ddots & \vdots \\ & & & r_{kk} \end{bmatrix}$$

Further, we get

$$[\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_k] = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_k] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ & r_{22} & \cdots & r_{2k} \\ & & \ddots & \vdots \\ & & & r_{kk} \end{bmatrix} \tag{12}$$

Expanding (12) yields

$$\begin{cases} \boldsymbol{h}_1 = \boldsymbol{q}_1 r_{11} & (a) \\ \boldsymbol{h}_2 = \boldsymbol{q}_1 r_{12} + \boldsymbol{q}_2 r_{22} & (b) \\ \vdots \\ \boldsymbol{h}_k = \boldsymbol{q}_1 r_{1k} + \boldsymbol{q}_2 r_{2k} + \cdots + \boldsymbol{q}_k r_{kk} & (c) \end{cases} \tag{13}$$

Since $\boldsymbol{Q}_k^\top\boldsymbol{Q}_k = \boldsymbol{I}$, then

$$\boldsymbol{q}_i^\top\boldsymbol{q}_j = 1 \quad \text{for} \quad i = j \quad \text{and} \quad \boldsymbol{q}_i^\top\boldsymbol{q}_j = 0 \quad \text{for} \quad i \neq j \tag{14}$$

From (13a), we have

$$\boldsymbol{h}_1^\top\boldsymbol{h}_1 = r_{11}\boldsymbol{q}_1^\top\boldsymbol{q}_1 r_{11} = r_{11}^2 \Rightarrow \begin{cases} r_{11} = \sqrt{\boldsymbol{h}_1^\top\boldsymbol{h}_1} \\ \boldsymbol{q}_1 = \boldsymbol{h}_1/r_{11} \end{cases} \tag{15}$$

For $1 \le i < k$,

$$\boldsymbol{q}_i^\top\boldsymbol{h}_k = \boldsymbol{q}_i^\top\boldsymbol{q}_i r_{ik} = r_{ik} \tag{16}$$

According to (13c), we denote

$$\tilde{\boldsymbol{h}}_k = \boldsymbol{q}_k r_{kk} = \boldsymbol{h}_k - \sum_{i=1}^{k-1}\boldsymbol{q}_i r_{ik} \tag{17}$$

so $r_{kk} = \sqrt{\tilde{\boldsymbol{h}}_k^\top\tilde{\boldsymbol{h}}_k}$ and $\boldsymbol{q}_k = \tilde{\boldsymbol{h}}_k/r_{kk}$.

When the $(k+1)$th hidden node is recruited, we get $\boldsymbol{H}_{k+1} = \left[\boldsymbol{H}_k, \boldsymbol{h}_{k+1}\right]$. Accordingly, the QR decomposition of $\boldsymbol{H}_{k+1}$ becomes

$$\boldsymbol{H}_{k+1} = \boldsymbol{Q}_{k+1}\boldsymbol{R}_{k+1} \tag{18}$$

where $\boldsymbol{Q}_{k+1} = \left[\boldsymbol{Q}_k, \boldsymbol{q}_{k+1}\right]$ and

$$\boldsymbol{R}_{k+1} = \begin{bmatrix} \boldsymbol{R}_k & \tilde{\boldsymbol{r}}_{k+1} \\ \boldsymbol{0} & r_{k+1,k+1} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1,k} & r_{1,k+1} \\ & r_{22} & \cdots & r_{2,k} & r_{2,k+1} \\ & & \ddots & \vdots & \vdots \\ & & & r_{kk} & r_{k,k+1} \\ & & & & r_{k+1,k+1} \end{bmatrix} \tag{19}$$

with $\tilde{\boldsymbol{r}}_{k+1} = \left[r_{1,k+1}, r_{2,k+1}, \cdots, r_{k,k+1}\right]^\top$. Together with (16), (17), (18), and (19), we have

$$\tilde{\boldsymbol{r}}_{k+1} = \boldsymbol{Q}_k^\top\boldsymbol{h}_{k+1} \tag{20}$$

$$\tilde{\boldsymbol{h}}_{k+1} = \boldsymbol{q}_{k+1}r_{k+1,k+1} = \boldsymbol{h}_{k+1} - \sum_{i=1}^{k}\boldsymbol{q}_i r_{i,k+1} = \boldsymbol{h}_{k+1} - \boldsymbol{Q}_k\tilde{\boldsymbol{r}}_{k+1} \tag{21}$$

$$r_{k+1,k+1} = \sqrt{\tilde{\boldsymbol{h}}_{k+1}^\top\tilde{\boldsymbol{h}}_{k+1}} \tag{22}$$

$$\boldsymbol{q}_{k+1} = \tilde{\boldsymbol{h}}_{k+1}/r_{k+1,k+1} \tag{23}$$

Using the following matrix identity [29]

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{D} \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{D}\boldsymbol{B}^{-1} \\ \boldsymbol{0} & \boldsymbol{B}^{-1} \end{bmatrix} \tag{24}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{D}$ are matrices of proper dimension, and $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible, hence

$$\boldsymbol{R}_{k+1}^{-1} = \begin{bmatrix} \boldsymbol{R}_k & \tilde{\boldsymbol{r}}_{k+1} \\ \boldsymbol{0} & r_{k+1,k+1} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{R}_k^{-1} & -\boldsymbol{R}_k^{-1}\tilde{\boldsymbol{r}}_{k+1}r_{k+1,k+1}^{-1} \\ \boldsymbol{0} & r_{k+1,k+1}^{-1} \end{bmatrix} \tag{25}$$

Finally,

$$\begin{aligned} \hat{\boldsymbol{\Theta}}_{k+1} &= \boldsymbol{H}_{k+1}^\dagger\boldsymbol{T} = \boldsymbol{R}_{k+1}^{-1}\boldsymbol{Q}_{k+1}^\top\boldsymbol{T} \\ &= \begin{bmatrix} \boldsymbol{R}_k^{-1} & -\boldsymbol{R}_k^{-1}\tilde{\boldsymbol{r}}_{k+1}r_{k+1,k+1}^{-1} \\ \boldsymbol{0} & r_{k+1,k+1}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{Q}_k^\top \\ \boldsymbol{q}_{k+1}^\top \end{bmatrix}\boldsymbol{T} \\ &= \begin{bmatrix} \hat{\boldsymbol{\Theta}}_k - \boldsymbol{R}_k^{-1}\tilde{\boldsymbol{r}}_{k+1}\hat{\boldsymbol{\theta}}_{k+1}^\top \\ \hat{\boldsymbol{\theta}}_{k+1}^\top \end{bmatrix} \end{aligned} \tag{26}$$

where $\hat{\boldsymbol{\theta}}_{k+1}^\top = \boldsymbol{q}_{k+1}^\top\boldsymbol{T}/r_{k+1,k+1}$. Notice that $\boldsymbol{R}_1^{-1} = 1/\sqrt{\boldsymbol{h}^\top\boldsymbol{h}}$ and $\hat{\boldsymbol{\Theta}}_1 = \boldsymbol{R}_1^{-1}\boldsymbol{q}_1^\top\boldsymbol{T}$, so both (25) and (26) can be computed incrementally with the increasing hidden nodes. When the output error is not more than the predefined level, i.e.,

$$\left\|\boldsymbol{H}_k\hat{\boldsymbol{\Theta}}_k - \boldsymbol{T}\right\|_F^2 \leqslant \epsilon \tag{27}$$

where $\epsilon > 0$ is the expected learning accuracy, QRI-ELM terminates. At each learning step, the total cost of (20)–(27) is $O(kNm)$. Successive k such updates incur a computational cost of $O(k^2Nm)$. The memory requirement of QRI-ELM is $O(kN)$.

4. GSI-ELM

For QRI-ELM, it recruits every hidden node, which is equivalent to randomly selecting one from a candidate set of infinite size. This mechanism without evaluation will lead to those hidden nodes recruited, which play a very minor role in the network output and thus increase the network complexity, which is not suitable for the testing time sensitive scenarios. To improve the quality of the hidden nodes recruited by GSI-ELM, an evaluating criterion can be added. It is impossible to recruit the *best* hidden node from a candidate set of infinite size at each learning step. However, a probabilistic trick can be exploited, which is to consider only a random subset of fixed size and selects the *best* hidden node from this set rather than performing an exhaustive search over a set of infinite size. This is a feasible strategy proved by Smola and Schölkopf [24]. In order to obtain an estimate that is with probability 0.95 among the best 0.05 of all estimates, a random set of size $\kappa = 59 = \left\lceil \frac{\log(0.05)}{\log(0.95)} \right\rceil$ will guarantee nearly as good performance as if we consider the whole set of hidden nodes.

4.1. A probabilistic trick

Assuming that these matrices $\boldsymbol{H}_k$, $\boldsymbol{R}_k$, $\boldsymbol{Q}_k$, and $\hat{\boldsymbol{\Theta}}_k$ have been obtained at the kth learning step, then an orthogonal projector [29] is defined as

$$\boldsymbol{P}_k = \boldsymbol{I} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top \tag{28}$$

with $\boldsymbol{P}_k^2 = \boldsymbol{P}_k$ and $\boldsymbol{P}_k = \boldsymbol{P}_k^\top$. Actually, $\boldsymbol{P}_k$ is the orthogonal projector onto the null space of the $\boldsymbol{H}_k$, because

$$\boldsymbol{P}_k = \boldsymbol{I} - \boldsymbol{H}_k\left(\boldsymbol{H}_k^\top\boldsymbol{H}_k\right)^{-1}\boldsymbol{H}_k^\top \tag{29}$$

Given that a random hidden nodes output matrix $\boldsymbol{H}_{\mathbb{B}}$, where $\mathbb{B} = \{1, 2, \dots, \kappa\}$, is generated at the $(k+1)$th learning step, then we can divide it as

$$\boldsymbol{H}_{\mathbb{B}} = \tilde{\boldsymbol{H}}_{\mathbb{B}} \oplus \tilde{\boldsymbol{H}}_{\mathbb{B}}^\perp \tag{30}$$

where $\oplus$ represents the direct sum operator [29],

$$\tilde{\boldsymbol{H}}_{\mathbb{B}} = \boldsymbol{P}_k\boldsymbol{H}_{\mathbb{B}} \tag{31}$$

and

$$\tilde{\boldsymbol{H}}_{\mathbb{B}}^\perp = (\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{H}_{\mathbb{B}} \tag{32}$$

Similarly, we get

$$\boldsymbol{T} = \tilde{\boldsymbol{T}}_k \oplus \tilde{\boldsymbol{T}}_k^\perp \tag{33}$$

where

$$\tilde{\boldsymbol{T}}_k = \boldsymbol{P}_k\boldsymbol{T} \tag{34}$$

$$\tilde{\boldsymbol{T}}_k^\perp = (\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{T} \tag{35}$$

and

$$\boldsymbol{H}_k = \tilde{\boldsymbol{H}}_k \oplus \tilde{\boldsymbol{H}}_k^\perp \tag{36}$$

where

$$\tilde{\boldsymbol{H}}_k = \boldsymbol{P}_k\boldsymbol{H}_k \tag{37}$$

$$\tilde{\boldsymbol{H}}_k^\perp = (\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{H}_k \tag{38}$$

Since $\tilde{\boldsymbol{H}}_k = \boldsymbol{P}_k\boldsymbol{H}_k = \left(\boldsymbol{I} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top\right)\boldsymbol{Q}_k\boldsymbol{R}_k = \boldsymbol{0}$, hence $\tilde{\boldsymbol{H}}_k^\perp = \boldsymbol{H}_k$. From (33), we know that the target $\boldsymbol{T}$ is divided into two parts. The part $\tilde{\boldsymbol{T}}_k^\perp$ in (35) can be represented very well using $\tilde{\boldsymbol{H}}_k^\perp$ in (38), because

$$\tilde{\boldsymbol{T}}_k^\perp = \tilde{\boldsymbol{H}}_k^\perp\hat{\boldsymbol{\Theta}}_k \tag{39}$$

That is to say,

$$\tilde{\boldsymbol{T}}_k^\perp = \boldsymbol{H}_k\hat{\boldsymbol{\Theta}}_k \tag{40}$$

Eq. (40) signifies that another part $\tilde{\boldsymbol{T}}_k$ can not be represented with $\boldsymbol{H}_k$ at all. In other words, $\tilde{\boldsymbol{T}}_k$ is the residual matrix for the kth learning step. In order to reduce $\tilde{\boldsymbol{T}}_k$ further, $\tilde{\boldsymbol{H}}_{\mathbb{B}}$ can be utilized to express it, because they lie in the same space, viz. the null space of the $\boldsymbol{H}_k$. Thus, an evaluating criterion is defined over $\tilde{\boldsymbol{H}}_{\mathbb{B}}$

$$\Delta_i = \frac{\left\|\tilde{\boldsymbol{h}}_i^\top\tilde{\boldsymbol{T}}_k\right\|_F^2}{\left\|\tilde{\boldsymbol{h}}_i\right\|_2} \tag{41}$$

where $\tilde{\boldsymbol{h}}_i \in \left\{\tilde{\boldsymbol{h}}_1, \tilde{\boldsymbol{h}}_2, \dots, \tilde{\boldsymbol{h}}_\kappa\right\}$, which are the corresponding columns of $\tilde{\boldsymbol{H}}_{\mathbb{B}}$. The larger is the Δ_i, the better the $\tilde{\boldsymbol{T}}_k$ is represented with $\tilde{\boldsymbol{h}}_i$. That is, $\tilde{\boldsymbol{h}}_i$ can incur the larger reduction on $\tilde{\boldsymbol{T}}_k$, which indicates that $\tilde{\boldsymbol{T}}_k$ can be represented using fewer hidden nodes and thus a more compact ELM is obtained. Hence, we can find the index of the hidden node to be recruited by

$$s = \arg\max_{i\in\mathbb{B}} \Delta_i \tag{42}$$

To obtain Δ_i, we need to compute both (31) and (34), which incur the computational costs of $O(N^2\kappa)$ and $O(N^2m)$, respectively. Moreover, the memory requirement is $O(N^2)$. To reduce the computational complexity of (41), we modify it as

$$\Delta_i = \frac{\left\|\tilde{\boldsymbol{h}}_i^\top\boldsymbol{T}\right\|_F^2}{\left\|\tilde{\boldsymbol{h}}_i\right\|_2} \tag{43}$$

Theorem 1. *Eq. (43) is equivalent to (41).*

Proof. According to (34),

$$\tilde{\boldsymbol{h}}_i^\top\tilde{\boldsymbol{T}} = \tilde{\boldsymbol{h}}_i^\top\boldsymbol{P}_k\boldsymbol{T} = \tilde{\boldsymbol{h}}_i^\top[\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{P}_k)]\boldsymbol{T} = \tilde{\boldsymbol{h}}_i^\top\boldsymbol{T} - \tilde{\boldsymbol{h}}_i^\top(\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{T} \tag{44}$$

Additionally,

$$\tilde{\boldsymbol{h}}_i^\top(\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{T} = \boldsymbol{h}_i^\top\boldsymbol{P}_k^\top(\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{T} \tag{45}$$

Plugging (28) into (45) yields

$$\boldsymbol{h}_i^\top\boldsymbol{P}_k^\top(\boldsymbol{I} - \boldsymbol{P}_k)\boldsymbol{T} = \boldsymbol{h}_i^\top\left(\boldsymbol{I} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top\right)\boldsymbol{Q}_k\boldsymbol{Q}_k^\top\boldsymbol{T} = \boldsymbol{0} \tag{46}$$

Thus $\tilde{\boldsymbol{h}}_i^\top\tilde{\boldsymbol{T}}_k = \tilde{\boldsymbol{h}}_i^\top\boldsymbol{T}$, our claim is finished. □

As thus, the cost of $O(N^2m)$ vanishes when Eq. (43) is utilized to calculate Δ_i. To cut down the computational complexity of calculating $\tilde{\boldsymbol{H}}_{\mathbb{B}}$ in (31), it is expanded as

$$\tilde{\boldsymbol{H}}_{\mathbb{B}} = \left(\boldsymbol{I} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top\right)\boldsymbol{H}_{\mathbb{B}} = \boldsymbol{H}_{\mathbb{B}} - \boldsymbol{Q}_k\boldsymbol{Q}_k^\top\boldsymbol{H}_{\mathbb{B}} \tag{47}$$

If $\boldsymbol{Q}_k^\top\boldsymbol{H}_{\mathbb{B}}$ in (47) is computed firstly rather than calculating $\boldsymbol{Q}_k^\top\boldsymbol{Q}_k$, the cost is reduced from $O(\max\{N^2k, N^2\kappa\})$ to $O(kN\kappa)$. Meanwhile, the memory cost is dropped to $O(\max\{Nk, N\kappa\})$. That is, the matrix $\boldsymbol{P}_k$ need not be calculated explicitly, and $\tilde{\boldsymbol{h}}_i (i \in \mathbb{B})$ is given using (47). When $\boldsymbol{h}_s$ is determined using (42), the updating strategy of QRI-ELM from (20) to (26) is employed to obtain $\hat{\boldsymbol{\Theta}}_{k+1}$. This procedure above is repeated until the stopping criterion is satisfied.

The stopping criterion of QRI-ELM can also be used to stop GSI-ELM. To curb the number of hidden nodes exactly, a positive integer $k_{\max}$ can be set as the stopping criterion. When k reaches $k_{\max}$, GSI-ELM stops recruiting hidden nodes.

4.2. The flowchart of GSI-ELM

See Algorithm 1.

Algorithm 1 GSI-ELM.

1. **input**: Input training data $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^N$ and activation function $h(\cdot)$; parameters κ, ϵ and $k_{\max}$.
2. **output**: $f_{\text{GSI-ELM}}(\boldsymbol{x}) = \sum_{i=1}^k \hat{\boldsymbol{\theta}}_i h(\boldsymbol{a}_i, b_i, \boldsymbol{x})$.
3. **initialize**:
   - Randomly generate $\boldsymbol{H}_{\mathbb{B}} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_\kappa]$, where $\mathbb{B} = \{1, 2, \ldots, \kappa\}$;
   - Calculate $\Delta_i = \frac{\|\boldsymbol{h}_i^\top \boldsymbol{T}\|_F^2}{\|\boldsymbol{h}_i\|_2}, i \in \mathbb{B}$; % $O(Nm\kappa)$
   - Choose $s = \arg\max_{i\in\mathbb{B}} \Delta_i$;
   - Calculate $r_{11} = \sqrt{\boldsymbol{h}_s^\top \boldsymbol{h}_s}$; % $O(N)$
   - Let $\boldsymbol{Q}_1 = \boldsymbol{h}_s / r_{11}$, $\boldsymbol{R}_1^{-1} = 1/r_{11}$, $\boldsymbol{H}_1 = \boldsymbol{h}_s$; % $O(N)$
   - Calculate $\hat{\boldsymbol{\Theta}}_1 = \boldsymbol{Q}_1 \boldsymbol{T}/r_{11}$; % $O(Nm)$
   - Let $\boldsymbol{A}_1 = \left[\boldsymbol{a}_s^\top, b_s\right]^\top$, where $\boldsymbol{a}_s$ and b_s are random parameters of $\boldsymbol{h}_s$, and k =1.
4. **while** $k < k_{\max}$ and $\left\|\boldsymbol{H}_k \hat{\boldsymbol{\Theta}}_k - \boldsymbol{T}\right\|_F^2 > \epsilon$ **do** % $O(Nmk)$
5. Randomly generate $\boldsymbol{H}_{\mathbb{B}} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$;
6. Obtain $\tilde{\boldsymbol{H}}_{\mathbb{B}}$ according to (47); % $O(kN\kappa)$
7. Obtain $\Delta_i (i \in \mathbb{B})$ according to (47); % $O(Nm\kappa)$
8. Find $\boldsymbol{h}_s$ and $\tilde{\boldsymbol{h}}_s$ according to (47);
9. Calculate $\tilde{\boldsymbol{r}}_{k+1} = \boldsymbol{Q}_k^\top \boldsymbol{h}_s$; % $O(kN)$
10. Calculate $r_{k+1,k+1} = \sqrt{\tilde{\boldsymbol{h}}_s^\top \tilde{\boldsymbol{h}}_s}$; % $O(N)$
11. Calculate $\boldsymbol{q}_{k+1} = \tilde{\boldsymbol{h}}_s / r_{k+1,k+1}$; % $O(N)$
12. Obtain $\boldsymbol{R}_{k+1}^{-1}$ according to (47); % $O(k^2)$
13. Obtain $\hat{\boldsymbol{\Theta}}_{k+1} = \left[\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_{k+1}\right]^\top$ according to (47); % $O(\max\{k^2, km\})$
14. Let $\boldsymbol{A}_{k+1} = \left[\boldsymbol{A}_k, [\boldsymbol{a}_s^\top, \boldsymbol{b}_s]^\top\right]$, where $\boldsymbol{a}_s$ and b_s are random parameters of $\boldsymbol{h}_s$;
15. Let $\boldsymbol{Q}_{k+1} = \left[\boldsymbol{Q}_k, \boldsymbol{q}_{k+1}\right]$, $\boldsymbol{H}_{k+1} = [\boldsymbol{H}_k, \boldsymbol{h}_s]$, and $k \leftarrow k+1$.
16. **end while**
17. **Return** $\hat{\boldsymbol{\Theta}}_k$ and $\boldsymbol{A}_k$.

4.3. Computational complexity of GSI-ELM

In GSI-ElM, the computational complexity of each row is listed behind the symbol %. Generally, both k and κ are larger than m. When GSI-ELM recruits one hidden node, the computational cost is $O(kN\kappa)$. Adding up these costs till k hidden nodes are recruited, we have the computational cost of $O(k^2N\kappa)$. In addition, the memory cost of GSI-ELM is $O(kN)$.

5. IGSI-ELM

In GSI-ELM, once one hidden node is recruited, it will not be discarded later. That is, the “nesting effect” exists in GSI-ELM. To solve this problem, some elimination mechanism can be introduced into GSI-ELM. Firstly, some evaluation criterion is defined. Then, based on this evaluation criterion, we can judge whether the existing hidden nodes are *good* or not. Finally, if the so-called *worst* hidden node is not the newly-recruited hidden node, it will be eliminated. Otherwise, no hidden nodes will be eliminated. That is, even though one hidden node is already recruited, it may be discarded later if its quality is not *good* enough. Hence, it is very crucial to define an appropriate evaluation criterion for IGSI-ELM.

5.1. The evaluation criterion

As known, ELM is equivalent to solving (6). Theoretically, if one hidden node is recruited, the objective value of (6) will decrease. Ideally, the objective value of (6) is dropped to zero, which, however, usually suffers from the overfitting phenomenon. Hence, we do not let the objective value of (6) equal zero. Intuitively, the more important the hidden node is, the larger reduction on the objective function it incurs.

Assume that at the $(k+1)$th learning step the hidden node $\boldsymbol{h}_s$, viz. $\boldsymbol{h}_{k+1}$, has already been determined via (42). In this situation, equation (6) becomes as

$$\min_{\boldsymbol{\Theta}_{k+1}} \left\{ J_{k+1} = \|\boldsymbol{H}_{k+1}\boldsymbol{\Theta}_{k+1} - \boldsymbol{T}\|_F^2 \right\} \tag{48}$$

If $\boldsymbol{h}_i (i = 1, \ldots, k+1)$ is eliminated from (48), it is denoted by

$$\min_{\boldsymbol{\Theta}_{k+1}^{(-i)}} \left\{ J_{k+1}^{(-i)} = \left\|\boldsymbol{H}_{k+1}^{(-i)}\boldsymbol{\Theta}_{k+1}^{(-i)} - \boldsymbol{T}\right\|_F^2 \right\} \tag{49}$$

Together with (48) and (49), an evaluation criterion over the ith hidden node is defined as

$$\delta_i = \hat{J}_{k+1}^{(-i)} - \hat{J}_{k+1}, \quad i \in \{1, \ldots, k+1\} \tag{50}$$

where $\hat{J}_{k+1}^{(-i)}$ and $\hat{J}_{k+1}$ represent the optimal objective values of (49) and (48), respectively. The larger the δ_i is, the more important the ith hidden node is. To obtain δ_i, if QR decomposition realized with Gram–Schmidt process is used to calculate $\hat{\boldsymbol{\Theta}}_{k+1}^{(-i)}$ as (11), the complexity cost is $O(Nk^2m)$[30], which amounts to implementing QRI-ELM one time. Hence, the total cost is up to $O(Nk^3m)$ in order to compute $\delta_i (i = 1, \ldots, k+1)$, meaning solving QRI-ELM k times, and this cost will become prohibitive as increasing k. Obviously, it is necessary to accelerate the computation of $\delta_i (i = 1, \ldots, k+1)$.

5.2. An accelerating scheme of δ_i

Theorem 2.

$$\delta_i = \frac{\left\|\hat{\boldsymbol{\theta}}_i\right\|_2^2}{\|\boldsymbol{p}_i\|_2^2}, \quad i = 1, \ldots, k+1 \tag{51}$$

holds, where $\hat{\boldsymbol{\theta}}_i^\top$ *is the ith row vector of* $\hat{\boldsymbol{\Theta}}_{k+1}$, $\boldsymbol{p}_i$ *is the ith column vector of* $\boldsymbol{R}_{k+1}^{-\top}$.

Proof. Based on (48), we can obtain the following optimization problem:

$$\min_{\boldsymbol{\Theta}_{k+1}^{\prime(i)}} \left\{ J_{k+1}^{\prime(i)} = \left\|\boldsymbol{H}_{k+1}\boldsymbol{\Theta}_{k+1}^{\prime(i)} - \boldsymbol{T}\right\|_F^2 + \left\|\boldsymbol{\lambda}^\top\boldsymbol{\Theta}_{k+1}^{\prime(i)}\right\|_F^2 \right\} \tag{52}$$

where all the entries of $\boldsymbol{\lambda} \in \Re^{(k+1)}$ are equal to zeros except $\boldsymbol{\lambda}_i = \lambda$, $\lambda > 0$. From (52), notice that

$$\hat{J}_{k+1}^{(-i)} = \lim_{\lambda\to\infty} \hat{J}_{k+1}^{\prime(i)} \tag{53}$$

where $\hat{J}_{k+1}^{\prime(i)}$ represents the optimal objective value of (52). Hence, Eq. (50) becomes

$$\delta_i = \lim_{\lambda\to\infty} \hat{J}_{k+1}^{\prime(i)} - \hat{J}_{k+1}, \quad i \in \{1, \ldots, k+1\} \tag{54}$$

On one side, according to (11), substituting $\hat{\boldsymbol{\Theta}}_{k+1} = \boldsymbol{R}_{k+1}^{-1}\boldsymbol{Q}_{k+1}^\top\boldsymbol{T}$ into (48) gets

$$\hat{J}_{k+1} = \mathrm{tr}\left(\boldsymbol{T}^\top\boldsymbol{T} - \boldsymbol{T}^\top\boldsymbol{Q}_{k+1}\boldsymbol{Q}_{k+1}^\top\boldsymbol{T}\right) \tag{55}$$

where $\mathrm{tr}(\cdot)$ represents the matrix trace. On the other side, letting $\frac{dJ_{k+1}^{\prime(i)}}{d\boldsymbol{\Theta}_{k+1}^{\prime(i)}} = 0$ generates

$$\hat{\boldsymbol{\Theta}}_{k+1}^{\prime(i)} = \left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1} + \boldsymbol{\lambda}\boldsymbol{\lambda}^\top\right)^{-1}\boldsymbol{H}_{k+1}^\top\boldsymbol{T} \tag{56}$$

Plugging (56) into (52) yields

$$\hat{J}_{k+1}^{\prime(i)} = \mathrm{tr}\left(\boldsymbol{T}^\top\boldsymbol{T} - \boldsymbol{T}^\top\boldsymbol{H}_{k+1}\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1} + \boldsymbol{\lambda}\boldsymbol{\lambda}^\top\right)^{-1}\boldsymbol{H}_{k+1}^\top\boldsymbol{T}\right) \tag{57}$$

With Sherman–Morrison formula [29]

$$(\boldsymbol{A} + \boldsymbol{a}\boldsymbol{b}^\top)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{a}\boldsymbol{b}^\top\boldsymbol{A}^{-1}}{1 + \boldsymbol{b}^\top\boldsymbol{A}^{-1}\boldsymbol{a}} \tag{58}$$

where $\boldsymbol{A}$ is an inventible matrix, $\boldsymbol{a}$ and $\boldsymbol{b}$ are column vectors with proper dimension, thus

$$\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1} + \boldsymbol{\lambda}\boldsymbol{\lambda}^\top\right)^{-1} = \left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1} - \frac{\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{\lambda}\boldsymbol{\lambda}^\top\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}}{1 + \boldsymbol{\lambda}^\top\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{\lambda}} \tag{59}$$

Eq. (59) is substituted into (57), so

$$\hat{J}_{k+1}^{\prime(i)} = \mathrm{tr}\left(\boldsymbol{T}^\top\boldsymbol{T} - \boldsymbol{T}^\top\boldsymbol{H}_{k+1}\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{H}_{k+1}^\top\boldsymbol{T}\right) + \mathrm{tr}\left(\frac{\boldsymbol{T}^\top\boldsymbol{H}_{k+1}\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{\lambda}\boldsymbol{\lambda}^\top\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{H}_{k+1}^\top\boldsymbol{T}}{1 + \boldsymbol{\lambda}^\top\left(\boldsymbol{H}_{k+1}^\top\boldsymbol{H}_{k+1}\right)^{-1}\boldsymbol{\lambda}}\right) \tag{60}$$

Due to $\boldsymbol{H}_{k+1} = \boldsymbol{Q}_{k+1}\boldsymbol{R}_{k+1}$ and $\hat{\boldsymbol{\Theta}}_{k+1} = \boldsymbol{R}_{k+1}^{-1}\boldsymbol{Q}_{k+1}^\top\boldsymbol{T}$, Eq. (60) is simplified as

$$\hat{J}_{k+1}^{\prime(i)} = \mathrm{tr}\left(\boldsymbol{T}^\top\boldsymbol{T} - \boldsymbol{T}^\top\boldsymbol{Q}_{k+1}\boldsymbol{Q}_{k+1}^\top\boldsymbol{T}\right) + \mathrm{tr}\left(\frac{\hat{\boldsymbol{\Theta}}_{k+1}^\top\boldsymbol{\lambda}\boldsymbol{\lambda}^\top\hat{\boldsymbol{\Theta}}_{k+1}}{1 + \boldsymbol{\lambda}^\top\boldsymbol{R}_{k+1}^{-1}\boldsymbol{R}^{-\top}\boldsymbol{\lambda}}\right) \tag{61}$$

Together with (54), (55), and (61), we get

$$\delta_i = \lim_{\lambda\to\infty}\mathrm{tr}\left(\frac{\hat{\boldsymbol{\Theta}}_{k+1}^\top\boldsymbol{\lambda}\boldsymbol{\lambda}^\top\hat{\boldsymbol{\Theta}}_{k+1}}{1 + \boldsymbol{\lambda}^\top\boldsymbol{R}_{k+1}^{-1}\boldsymbol{R}^{-\top}\boldsymbol{\lambda}}\right) = \lim_{\lambda\to\infty}\frac{\lambda^2\left\|\hat{\boldsymbol{\theta}}_i\right\|_2^2}{1 + \lambda^2\|\boldsymbol{p}_i\|_2^2} = \frac{\left\|\hat{\boldsymbol{\theta}}_i\right\|_2^2}{\|\boldsymbol{p}_i\|_2^2} \tag{62}$$

Now, this proof is finished. □

At the $(k+1)$th learning step, $\hat{\boldsymbol{\Theta}}_{k+1}$ and $\boldsymbol{R}_{k+1}^{-1}$ have already been computed, so $\delta_i (i = 1, \ldots, k+1)$ can be easily got using (51) at a cost of $O(k^2)$. Compared with the computational cost, viz. $O(Nk^3m)$, of δ_i using (50) directly, this drop is very obvious. Additionally, $\delta_i \geq 0$ usually holds from (62). Since the evaluation criterion δ_i is got, in the following the elimination mechanism will be introduced.

5.3. The elimination mechanism

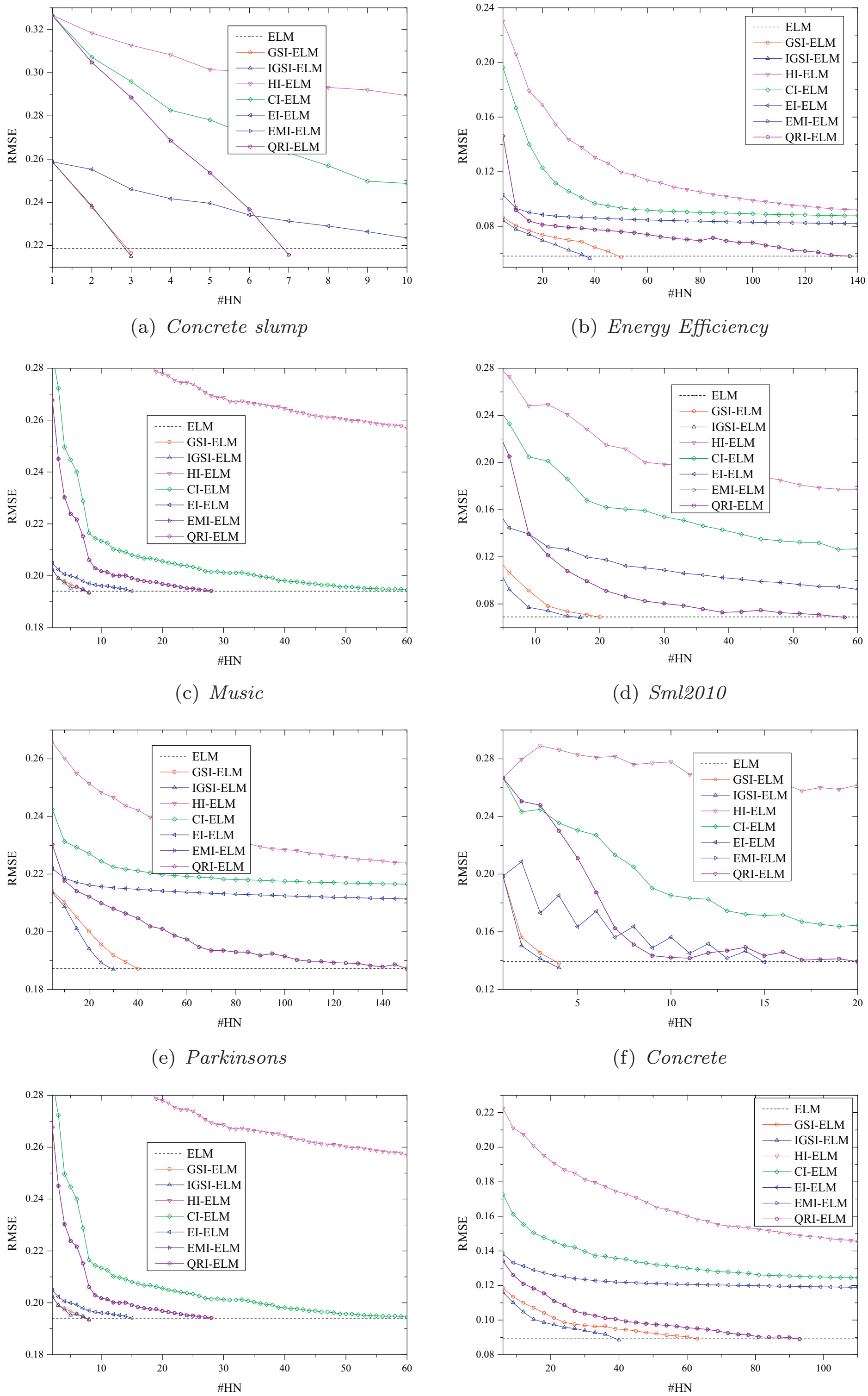
According to (51), one δ_i is defined for every hidden node at the $(k+1)$th learning step, so we can rank these hidden nodes recruited based on their importance. The larger the δ_i is, the more important its corresponding hidden node is. From this rank, the hidden node, represented by $\boldsymbol{h}_{i_{\min}} (1 \leqslant r_{\min} \leqslant k+1)$, holding the least $\delta_{i_{\min}}$ is found. Then, two cases are encountered:

(i) $r_{\min} = k+1$
For this case, the newly-recruited hidden node is the least important one. That is to say, the previously-recruited hidden nodes are *good* enough. The objective value of (6) can be reduced further via retaining the $(k+1)$th hidden node.

Table 1
Description of data sets.

Data sets	#Training	#Testing	#Inputs	#Outputs	#Classes
Concrete slump	57	46	7	3	–
Energy efficiency	422	346	8	2	–
Music	582	477	68	2	–
Sml2010	2275	1862	16	2	–
Parkinsons	3231	2644	18	2	–
Concrete	553	452	8	1	–
Airfoil	827	676	5	1	–
Winequality white	2179	1782	11	1	–
Abalone	2297	1880	8	1	–
Cpu_small	4506	3686	12	1	–
Kinematics	4506	3686	8	1	–
Delta_ailerons	3921	3208	5	1	–
Delta_elevators	5234	4283	6	1	–
Iris	81	64	4	–	3
Ionosphere	193	157	33	–	2
Balance	344	281	4	–	3
Vehicle	465	381	18	–	4
Hill_valley	667	545	100	–	2
Yeast	698	571	9	–	4
Banknote	741	607	4	–	2
Car	950	778	6	–	4
Statlog	1147	939	18	–	7
Waveform	2750	2250	40	–	3
Waveform2	2750	2250	40	–	2
Landsat	3539	2896	36	–	6
Mushroom	3843	3145	20	–	2

Notes: #Training represents the number of training data, #Testing represents the number of testing data, #Inputs represents the number of input attributes, #Outputs represents the number of output targets (regression applications), #Classes represents the number of classes (classification applications).



(a) *Concrete slump* (b) *Energy Efficiency*

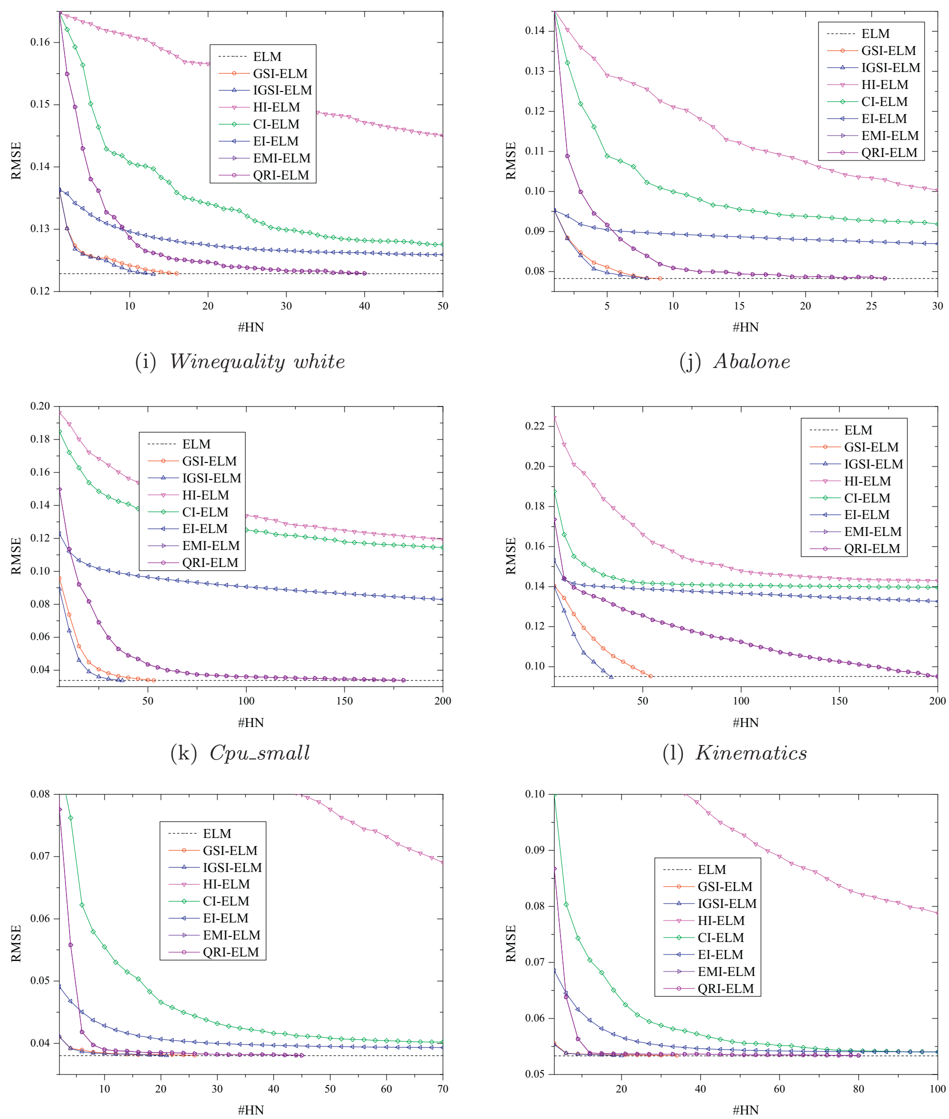
(c) *Music* (d) *Sml2010*

(e) *Parkinsons* (f) *Concrete*

(g) *Music* (h) *Airfoil*

Fig. 1. Comparisons of the RMSE in terms of #HN for different algorithms on regression applications.

(ii) $r_{\min} \neq k+1$
This case means that the $(k+1)$th hidden node is better than the $r_{\min}$th one. In this situation, we can eliminate the $r_{\min}$th hidden node, which is equivalent to replacing $\boldsymbol{h}_{r_{\min}}$ with $\boldsymbol{h}_{k+1}$. Due to $\hat{J}_{k+1}^{(-(k+1))} > \hat{J}_{k+1}^{(-r_{\min})}$, the objective value $\hat{J}$ continues decreasing. This behavior keeps the number of hidden nodes constant and meanwhile decreases the objective value, which obeys the principle of Occam's razor "plurality must never be posited with necessity" [28] and improves the ELM performance. After eliminating the $i_{\min}$th



(i) *Winequality white* (j) *Abalone*

(k) *Cpu_small* (l) *Kinematics*

(m) *Delta_ailerons* (n) *Delta_elevators*

Fig. 1. Continued

hidden node, we need update $\boldsymbol{R}^{-1}$, $\boldsymbol{Q}$, $\boldsymbol{H}$, and $\hat{\boldsymbol{\Theta}}$. If $r_{\min} \neq 1$, the partial strategy can be utilized. That is, let

$$\boldsymbol{R}^{-1}_{r_{\min}-1} \leftarrow \boldsymbol{R}^{-1}_{k+1}(1 \sim r_{\min}-1, 1 \sim r_{\min}-1) \tag{63}$$

$$\boldsymbol{Q}_{r_{\min}-1} \leftarrow \boldsymbol{Q}_{k+1}(\cdot, 1 \sim r_{\min}-1) \tag{64}$$

$$\boldsymbol{H}_{r_{\min}-1} \leftarrow \boldsymbol{H}_{k+1}(\cdot, 1 \sim r_{\min}-1) \tag{65}$$

and

$$\hat{\boldsymbol{\Theta}}_{r_{\min}-1} = \boldsymbol{R}^{-1}_{r_{\min}-1}\boldsymbol{Q}^{\top}_{r_{\min}-1}\boldsymbol{T} \tag{66}$$

where $1 \sim r_{\min}-1$ denotes columns or rows from 1 to $r_{\min}-1$, $\cdot$ denotes all the columns or rows. From (63), (64), and (65), we know that $\boldsymbol{Q}_{r_{\min}-1}\boldsymbol{R}_{r_{\min}-1} = \boldsymbol{H}_{r_{\min}-1}$ satisfies the QR decomposition, where $\boldsymbol{Q}^{\top}_{r_{\min}-1}\boldsymbol{Q}_{r_{\min}-1} = \boldsymbol{I}$ and $\boldsymbol{R}_{r_{\min}-1}$ is an upper triangular matrix. Based on $\boldsymbol{R}^{-1}_{r_{\min}-1}$, $\boldsymbol{Q}_{r_{\min}-1}$, and $\hat{\boldsymbol{\Theta}}_{r_{\min}-1}$, when the columns of $\boldsymbol{H}_{k+1}$ from $r_{\min}+1 \sim k+1$ are sequentially recruited, the matrices $\boldsymbol{R}^{-1}_{k}$, $\boldsymbol{Q}_k$, and $\hat{\boldsymbol{\Theta}}_k$ can be obtained using the incremental updating strategy from (20) to (26). This partial strategy can avoid implementing the QR decomposition from scratch, which drops the computational cost. Otherwise, we have to perform a full QR decomposition from the beginning to obtain $\boldsymbol{R}^{-1}_{k}$, $\boldsymbol{Q}_k$, and $\hat{\boldsymbol{\Theta}}_k$ for the case $r_{\min} = 1$. Finally, let

$$\boldsymbol{H}_k \leftarrow \boldsymbol{H}_{k+1}(\cdot, -r_{\min}) \tag{67}$$

where $-r_{\min}$ represents the $r_{\min}$th column or row eliminated.

When this elimination mechanism is finished, the procedure starts to recruit next hidden node or terminates.

5.4. The flowchart of IGSI-ELM

See Algorithm 2.

5.5. Computational complexity of IGSI-ELM

Compared with GSI-ELM, IGSI-ELM requires an additional cost, viz. $O(k^2(N-r_{\min}))$, of the elimination mechanism. If the elimination mechanism is implemented l times on average for one hidden

Algorithm 2 IGSI-ELM.

input: Input training data $\{(\boldsymbol{x}_i, \boldsymbol{t}_i)\}_{i=1}^N$ and activation function $h(\cdot)$; parameters κ, ϵ and k_{max}.

output: $f_{\text{IGSI-ELM}}(\boldsymbol{x}) = \sum_{i=1}^k \hat{\boldsymbol{\theta}}_i h(\boldsymbol{a}_i, b_i, \boldsymbol{x})$.

initialize:

- Randomly generate $\boldsymbol{H}_{\mathbb{B}} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_\kappa]$, where $\mathbb{B} = \{1, 2, \ldots, \kappa\}$;
- Calculate $\Delta_i = \frac{\|\boldsymbol{h}_i^\top \boldsymbol{T}\|_F^2}{\|\boldsymbol{h}_i\|_2}, i \in \mathbb{B}$; % $O(Nm\kappa)$
- Choose $s = \arg\max_{i\in\mathbb{B}} \Delta_i$;
- Calculate $r_{11} = \sqrt{\boldsymbol{h}_s^\top \boldsymbol{h}_s}$; % $O(N)$
- Let $\boldsymbol{Q}_1 = \boldsymbol{h}_s / r_{11}$, $\boldsymbol{R}_1^{-1} = 1/r_{11}$, $\boldsymbol{H}_1 = \boldsymbol{h}_s$; % $O(N)$
- Calculate $\hat{\boldsymbol{\Theta}}_1 = \boldsymbol{Q}_1 \boldsymbol{T} / r_{11}$; % $O(Nm)$
- Let $\boldsymbol{A}_1 = \left[\boldsymbol{a}_s^\top, b_s\right]^\top$, where $\boldsymbol{a}_s$ and b_s are random parameters of $\boldsymbol{h}_s$, and $k = 1$.

while $k < k_{max}$ and $\left\|\boldsymbol{H}_k \hat{\boldsymbol{\Theta}}_k - \boldsymbol{T}\right\|_F^2 > \epsilon$ **do** % $O(Nmk)$
  Randomly generate $\boldsymbol{H}_{\mathbb{B}} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k]$;
  Obtain $\tilde{\boldsymbol{H}}_{\mathbb{B}}$ according to (67); % $O(kN\kappa)$
  Obtain $\Delta_i (i \in \mathbb{B})$ according to (67); % $O(Nm\kappa)$
  Find $\boldsymbol{h}_s$ and $\tilde{\boldsymbol{h}}_s$ according to (67);
  Calculate $\tilde{\boldsymbol{r}}_{k+1} = \boldsymbol{Q}_k^\top \boldsymbol{h}_s$; % $O(kN)$
  Calculate $r_{k+1,k+1} = \sqrt{\tilde{\boldsymbol{h}}_s^\top \tilde{\boldsymbol{h}}_s}$; % $O(N)$
  Calculate $\boldsymbol{q}_{k+1} = \tilde{\boldsymbol{h}}_s / r_{k+1,k+1}$; % $O(N)$
  Obtain $\boldsymbol{R}_{k+1}^{-1}$ according to (67); % $O(k^2)$
  Obtain $\hat{\boldsymbol{\Theta}}_{k+1} = \left[\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_{k+1}\right]^\top$ according to (67); % $O(\max\{k^2, km\})$
  Let $\boldsymbol{A}_{k+1} = \left[\boldsymbol{A}_k, [\boldsymbol{a}_s^\top, \boldsymbol{b}_s]^\top\right]$, where $\boldsymbol{a}_s$ and b_s are random parameters of $\boldsymbol{h}_s$;
  Let $\boldsymbol{Q}_{k+1} = \left[\boldsymbol{Q}_k, \boldsymbol{q}_{k+1}\right]$, and $\boldsymbol{H}_{k+1} = [\boldsymbol{H}_k, \boldsymbol{h}_s]$;
  Calculate $\delta_i (i = 1, \ldots, k+1)$ according to (67); % $O(k)$
  Find out $\boldsymbol{h}_{i_{\min}}$ corresponding to $\delta_{i_{\min}}$;
  if $i_{\min} \neq k+1$
    if $i_{\min} \neq 1$
      Obtain $\boldsymbol{R}_{r_{\min}-1}^{-1}$, $\boldsymbol{Q}_{r_{\min}-1}$, and $\hat{\boldsymbol{\Theta}}_{r_{\min}-1}$ from (67), (67), and (67), respectively;
      Obtain $\boldsymbol{R}_k^{-1}$, $\boldsymbol{Q}_k$, and $\hat{\boldsymbol{\Theta}}_k$ using the incremental updating strategy from (67) to (67); % $O(k^2(N - r_{\min}))$
    else
      Obtain $\boldsymbol{R}_k^{-1}$, $\boldsymbol{Q}_k$, and $\hat{\boldsymbol{\Theta}}_k$ with a full QR decomposition from scratch; % $O(k^2 N)$
    end if
    Let $\boldsymbol{H}_k \leftarrow \boldsymbol{H}_{k+1}(\cdot, -r_{\min})$, and $\boldsymbol{A}_k \leftarrow \boldsymbol{A}_{k+1}(\cdot, -r_{\min})$;
  else
    Let $k \leftarrow k+1$;
  end if
end while
Return $\hat{\boldsymbol{\Theta}}_k$ and $\boldsymbol{A}_k$.

node recruited, then this additional cost is $O(k^3 l(N - r_{\min}))$ after recruiting k hidden nodes. Generally, IGSI-ELM usually needs fewer hidden nodes than GSI-ELM when they reach nearly the same generalization performance. That is, the additional cost required is usually less than $O(k^3 l(N - r_{\min}))$. The memory requirement of IGSI-ELM is the same as that of GSI-ELM, viz. $O(Nk)$.

6. Experiments

In this section, we provide experiments conducted in order to illustrate the effectiveness and feasibility of the proposed GSI-ELM and IGSI-ELM via comparison with the algorithms aforementioned, viz. HI-ELM, CI-ELM, EI-ELM, EMI-ELM, and QRI-ELM. All experiments have been carried out on a personal desktop with Intel®Core™ i7-6600U CPU 2.60 GHz processor, 8.00 GB memory, and Windows 10 operating system in MATLAB2016a environment. In this paper, twenty-six benchmark data sets listed in Table 1 are utilized to perform experiments, which consist of thirteen regression applications (*Concrete slump*, *Energy efficiency*, *Music*, *Sml2010*, *Parkinsons*, *Concrete*, *Airfoil*, *Winequality white*, *Abalone*, *Cpu_small*, *Kinematics*, *Delta_ailerons*, and *Delta_elevators*) and thirteen classification applications including *Iris*, *Ionosphere*, *Balance*, *Vehicle*, *Hill_valley*, *Yeast*, *Banknote*, *Car*, *Statlog*, *Waveform*, *Waveform2*, *Landsat*, and *Mushroom*. Thereinto, *Cpu_small*, *Kinematics*, *Delta_ailerons*, and *Delta_elevators* are available from the

Table 2
Performance comparison on regression applications.

Data sets	Algorithms	L	RMSE	Training time (s)	Testing time (s)
Concrete slump	ELM	10	2.187e−01 ± 1.430e−02	**0.000 ± 0.000**	0.000 ± 0.000
	GSI-ELM	**3**	2.167e−01 ± 1.184e−02	0.002 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**3**	2.150e−01 ± 1.682e−02	0.003 ± 0.002	0.000 ± 0.000
	HI-ELM	10	2.894e−01 ± 2.684e−02	0.001 ± 0.000	0.000 ± 0.000
	CI-ELM	10	2.487e−01 ± 2.395e−02	0.001 ± 0.000	0.000 ± 0.000
	EI-ELM	10	2.235e−01 ± 1.029e−02	0.008 ± 0.002	0.000 ± 0.000
	EMI-ELM	7	2.158e−01 ± 1.025e−02	0.001 ± 0.000	0.000 ± 0.000
	QRI-ELM	7	2.158e−01 ± 1.025e−02	0.001 ± 0.000	0.000 ± 0.000
Energy efficiency	ELM	140	5.815e−02 ± 3.806e−03	**0.005 ± 0.001**	0.001 ± 0.000
	GSI-ELM	50	5.716e−02 ± 3.219e−03	0.039 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**38**	5.664e−02 ± 5.776e−03	0.111 ± 0.011	0.000 ± 0.000
	HI-ELM	140	9.196e−02 ± 7.208e−03	0.007 ± 0.000	0.001 ± 0.000
	CI-ELM	140	8.762e−02 ± 3.219e−03	0.010 ± 0.001	0.001 ± 0.000
	EI-ELM	140	8.195e−02 ± 1.591e−03	0.159 ± 0.003	0.001 ± 0.000
	EMI-ELM	137	5.804e−02 ± 4.254e−03	0.517 ± 0.006	0.001 ± 0.000
	QRI-ELM	137	5.804e−02 ± 4.254e−03	0.027 ± 0.001	0.001 ± 0.000
Music	ELM	60	1.941e−01 ± 2.852e−03	**0.003 ± 0.001**	0.001 ± 0.000
	GSI-ELM	**8**	1.934e−01 ± 2.815e−03	0.013 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**8**	1.936e−01 ± 2.230e−03	0.021 ± 0.003	0.000 ± 0.000
	HI-ELM	60	2.569e−01 ± 5.647e−02	0.008 ± 0.001	0.001 ± 0.000
	CI-ELM	60	1.946e−01 ± 2.146e−03	0.009 ± 0.000	0.001 ± 0.000
	EI-ELM	15	1.941e−01 ± 3.904e−03	0.028 ± 0.001	0.000 ± 0.000
	EMI-ELM	28	1.942e−01 ± 2.820e−03	0.171 ± 0.002	0.000 ± 0.000
	QRI-ELM	28	1.942e−01 ± 2.820e−03	0.006 ± 0.000	0.000 ± 0.000
Sml2010	ELM	60	6.906e−02 ± 6.264e−03	**0.009 ± 0.001**	0.004 ± 0.001
	GSI-ELM	20	6.895e−02 ± 2.395e−03	0.083 ± 0.002	0.001 ± 0.000
	IGSI-ELM	**17**	6.848e−02 ± 7.899e−03	0.157 ± 0.026	0.001 ± 0.000
	HI-ELM	60	1.774e−01 ± 3.148e−02	0.011 ± 0.001	0.003 ± 0.000
	CI-ELM	60	1.267e−01 ± 2.803e−02	0.014 ± 0.001	0.003 ± 0.000
	EI-ELM	60	9.249e−02 ± 1.360e−02	0.260 ± 0.001	0.003 ± 0.000
	EMI-ELM	58	6.864e−02 ± 6.628e−03	4.946 ± 0.013	0.003 ± 0.000
	QRI-ELM	58	6.864e−02 ± 6.628e−03	0.024 ± 0.001	0.003 ± 0.000
Parkinsons	ELM	150	1.872e−01 ± 7.203e−03	0.030 ± 0.001	0.012 ± 0.001
	GSI-ELM	40	1.871e−01 ± 1.959e−03	0.249 ± 0.004	0.003 ± 0.000
	IGSI-ELM	**30**	1.868e−01 ± 1.863e−03	0.571 ± 0.053	0.002 ± 0.000
	HI-ELM	150	2.238e−01 ± 4.842e−03	**0.028 ± 0.001**	0.012 ± 0.000
	CI-ELM	150	2.165e−01 ± 1.084e−03	0.041 ± 0.001	0.012 ± 0.000
	EI-ELM	150	2.114e−01 ± 3.803e−04	1.034 ± 0.010	0.012 ± 0.000
	EMI-ELM	150	1.872e−01 ± 7.203e−03	31.749 ± 0.269	0.012 ± 0.000
	QRI-ELM	150	1.872e−01 ± 7.203e−03	0.336 ± 0.002	0.012 ± 0.001
Concrete	ELM	20	1.392e−01 ± 1.716e−02	**0.001 ± 0.000**	0.000 ± 0.000
	GSI-ELM	**4**	1.381e−01 ± 1.100e−02	0.005 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**4**	1.350e−01 ± 9.892e−03	0.007 ± 0.003	0.000 ± 0.000
	HI-ELM	20	2.619e−01 ± 6.009e−02	**0.001 ± 0.000**	0.000 ± 0.000
	CI-ELM	20	1.646e−01 ± 1.948e−02	0.002 ± 0.000	0.000 ± 0.000
	EI-ELM	15	1.389e−01 ± 5.640e−03	0.021 ± 0.001	0.000 ± 0.000
	EMI-ELM	20	1.392e−01 ± 1.716e−02	0.106 ± 0.001	0.000 ± 0.000
	QRI-ELM	20	1.392e−01 ± 1.716e−02	0.002 ± 0.000	0.000 ± 0.000
Airfoil	ELM	110	8.914e−02 ± 4.255e−03	**0.006 ± 0.001**	0.002 ± 0.000
	GSI-ELM	63	8.915e−02 ± 1.887e−03	0.079 ± 0.005	0.001 ± 0.000
	IGSI-ELM	**40**	8.850e−02 ± 1.413e−03	0.192 ± 0.023	0.001 ± 0.000
	HI-ELM	110	1.453e−01 ± 1.533e−02	**0.006 ± 0.001**	0.002 ± 0.000
	CI-ELM	110	1.245e−01 ± 1.631e−03	0.008 ± 0.001	0.002 ± 0.000
	EI-ELM	110	1.189e−01 ± 6.577e−04	0.150 ± 0.005	0.001 ± 0.000
	EMI-ELM	93	8.913e−02 ± 2.849e−03	1.265 ± 0.005	0.001 ± 0.000
	QRI-ELM	93	8.913e−02 ± 2.850e−03	0.022 ± 0.001	0.001 ± 0.000
Winequality white	ELM	50	1.229e−01 ± 5.149e−04	**0.006 ± 0.001**	0.003 ± 0.001
	GSI-ELM	16	1.229e−01 ± 5.175e−04	0.046 ± 0.003	0.001 ± 0.000
	IGSI-ELM	**13**	1.227e−01 ± 8.735e−04	0.080 ± 0.010	0.001 ± 0.000
	HI-ELM	50	1.451e−01 ± 8.055e−03	0.007 ± 0.000	0.002 ± 0.000
	CI-ELM	50	1.276e−01 ± 1.454e−03	0.009 ± 0.001	0.002 ± 0.000
	EI-ELM	50	1.259e−01 ± 5.406e−04	0.151 ± 0.005	0.002 ± 0.000
	EMI-ELM	40	1.229e−01 ± 5.692e−04	3.154 ± 0.007	0.002 ± 0.000
	QRI-ELM	40	1.229e−01 ± 5.692e−04	0.015 ± 0.001	0.002 ± 0.000
Abalone	ELM	30	7.824e−02 ± 1.249e−03	**0.004 ± 0.000**	0.002 ± 0.001
	GSI-ELM	9	7.827e−02 ± 6.384e−04	0.040 ± 0.002	0.000 ± 0.000
	IGSI-ELM	**8**	7.824e−02 ± 7.812e−04	0.052 ± 0.010	0.000 ± 0.000
	HI-ELM	30	1.003e−01 ± 6.933e−03	**0.004 ± 0.000**	0.001 ± 0.000
	CI-ELM	30	9.182e−02 ± 2.896e−03	0.005 ± 0.000	0.001 ± 0.000

(*continued on next page*)

Table 2 (*continued*)

Data sets	Algorithms	L	RMSE	Training time (s)	Testing time (s)
	EI-ELM	30	8.696e−02 ± 1.776e−03	0.109 ± 0.002	0.001 ± 0.000
	EMI-ELM	26	7.828e−02 ± 1.227e−03	2.181 ± 0.006	0.001 ± 0.000
	QRI-ELM	26	7.828e−02 ± 1.227e−03	0.008 ± 0.002	0.001 ± 0.000
Cpu_small	ELM	200	3.382e−02 ± 8.900e−04	0.082 ± 0.002	0.016 ± 0.001
	GSI-ELM	53	3.383e−02 ± 4.442e−04	0.476 ± 0.006	0.005 ± 0.000
	IGSI-ELM	**37**	3.372e−02 ± 4.171e−04	1.072 ± 0.106	0.004 ± 0.000
	HI-ELM	200	1.194e−01 ± 7.156e−03	**0.038 ± 0.002**	0.016 ± 0.001
	CI-ELM	200	1.145e−01 ± 3.163e−03	0.048 ± 0.001	0.016 ± 0.001
	EI-ELM	200	8.293e−02 ± 1.362e−03	1.561 ± 0.013	0.016 ± 0.001
	EMI-ELM	180	3.389e−02 ± 1.170e−03	73.982 ± 0.537	0.016 ± 0.001
	QRI-ELM	180	3.389e−02 ± 1.170e−03	0.728 ± 0.062	0.015 ± 0.001
Kinematics	ELM	200	9.513e−02 ± 2.508e−03	0.078 ± 0.003	0.017 ± 0.001
	GSI-ELM	54	9.519e−02 ± 2.354e−03	0.489 ± 0.005	0.004 ± 0.000
	IGSI-ELM	**34**	9.473e−02 ± 1.715e−03	1.052 ± 0.121	0.003 ± 0.000
	HI-ELM	200	1.430e−01 ± 3.046e−03	**0.035 ± 0.001**	0.017 ± 0.001
	CI-ELM	200	1.395e−01 ± 1.166e−03	0.045 ± 0.002	0.016 ± 0.001
	EI-ELM	200	1.327e−01 ± 1.060e−03	1.571 ± 0.009	0.015 ± 0.002
	EMI-ELM	200	9.513e−02 ± 2.508e−03	86.685 ± 0.145	0.016 ± 0.002
	QRI-ELM	200	9.513e−02 ± 2.508e−03	0.859 ± 0.003	0.016 ± 0.002
Delta_ailerons	ELM	70	3.802e−02 ± 9.332e-05	0.016 ± 0.001	0.007 ± 0.001
	GSI-ELM	26	3.808e-02 ± 1.324e-04	0.174 ± 0.004	0.002 ± 0.000
	IGSI-ELM	**21**	3.804e-02 ± 9.398e-05	0.373 ± 0.051	0.001 ± 0.000
	HI-ELM	70	6.903e-02 ± 1.490e-02	**0.011 ± 0.001**	0.006 ± 0.000
	CI-ELM	70	4.021e-02 ± 8.805e-04	0.015 ± 0.002	0.006 ± 0.000
	EI-ELM	70	3.932e-02 ± 2.208e-04	0.442 ± 0.004	0.005 ± 0.001
	EMI-ELM	45	3.806e-02 ± 1.491e-04	11.280 ± 0.024	0.005 ± 0.001
	QRI-ELM	45	3.806e-02 ± 1.491e-04	0.036 ± 0.002	0.004 ± 0.000
Delta_elevators	ELM	100	5.333e-02 ± 1.037e-04	0.029 ± 0.001	0.012 ± 0.001
	GSI-ELM	34	5.339e-02 ± 1.053e-04	0.329 ± 0.004	0.003 ± 0.000
	IGSI-ELM	**20**	5.336e-02 ± 7.767e-05	0.468 ± 0.049	0.002 ± 0.000
	HI-ELM	100	7.880e-02 ± 8.276e-03	**0.018 ± 0.001**	0.012 ± 0.000
	CI-ELM	100	5.403e-02 ± 3.401e-04	0.024 ± 0.001	0.011 ± 0.000
	EI-ELM	100	5.402e-02 ± 2.373e-04	0.843 ± 0.004	0.011 ± 0.000
	EMI-ELM	80	5.338e-02 ± 1.331e-04	38.018 ± 0.185	0.010 ± 0.000
	QRI-ELM	80	5.338e-02 ± 1.331e-04	0.154 ± 0.003	0.010 ± 0.000

data collection[1]. The rest are obtained from the well-known UCI machine learning repository[2]. For each data set, it is divided into two subsets, viz. the training set (about 55%) and the testing set (about 45%), whose details are described in Table 1, and its inputs (attributes) have been normalized into the range [−1, 1].

In regression applications, their outputs (targets) have been normalized into [0, 1], and one performance index, i.e., the rooted mean squared errors (RMSE), is defined as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i}^{\#\mathrm{Testing}} \left\| \hat{\boldsymbol{T}}_i - \boldsymbol{T}_i \right\|_F^2}{\#\mathrm{Testing} \times m}} \tag{68}$$

where $\hat{\boldsymbol{T}}_i$ denotes the prediction of the desired $\boldsymbol{T}_i$, #Testing is the total number of the testing data. A smaller RMSE usually means a better generalization performance for an algorithm.

On the contrary, the higher prediction accuracy (Acc) indicates a superior algorithm with respect to the generalization performance for classification applications. m-class classifiers have m output nodes. If the original class label is p, the expected output vector of the m output nodes is $\boldsymbol{t}_i = [-1, \ldots, -1, \overset{p}{1}, -1, \ldots, -1]^\top$. In this case, only the pth entry of $\boldsymbol{t}_i$ is one, while the rest entries are set to −1.

In this paper, the sigmoidal $h(\boldsymbol{a}, b, \boldsymbol{x}) = 1/\left(1 + \exp\left(\boldsymbol{a}^\top \boldsymbol{x} + b\right)\right)$ is chosen as activation function for all the algorithms, where the input weight $\boldsymbol{a}$ and bias b are randomly chosen from the range [-1, 1]. For the traditional ELM, the number of hidden nodes L is decided from the set $\{10, 20, \ldots, 200\}$ using cross validation technique [31]. To obtain the robust statistical results, all experiments are averaged over thirty different random runs.

6.1. Regression applications

The experimental results are demonstrated in Fig. 1. In each panel, the dash line generated by the traditional ELM is chosen as the benchmark line. The RMSEs decrease with increasing the number of hidden nodes (#HN). At first the RMSE lines drop fast and gradually change slowly. HI-ELM is the first incremental learning algorithm. Its convergence rate is the lowest. The main reason is that HI-ELM fixes the output weights of all the existing nodes when a new node is recruited. Hence, CI-ELM improves the convergence rate of HI-ELM via recomputing the output weights of the existing hidden nodes using a convex optimization during the process of recruiting hidden nodes. EI-ELM is an enhanced version of HI-ELM, in which the hidden node incurring the largest residual error decrease is recruited from a candidate set. In our experiments, the size of the candidate set for EI-ELM is the same as that for GSI-ELM and IGSI-ELM, say, $\kappa = 59$. Compared with HI-ELM and CI-ELM, EI-ELM achieves the fastest convergence rate. However, HI-ELM, CI-ELM, and EI-ELM do not touch the dash line when they need the same #HN as the traditional ELM.

The lines of both EMI-ELM and QRI-ELM are overlapped. Compared with HI-ELM, CI-ELM, and EI-ELM, QRI-ELM/EMI-ELM usually needs less #HN when reaching the dash line. That is to say, QRI-ELM/EMI-ELM owns the priority in terms of #HN. However, QRI-ELM/EMI-ELM loses this advantage over GSI-ELM and IGSI-ELM. The line of IGSI-ELM is usually lower than that of GSI-ELM, which

[1] http://www.dcc.fc.up.pt/%7Eltorgo/Regression/DataSets.html.
[2] http://archive.ics.uci.edu/ml/.

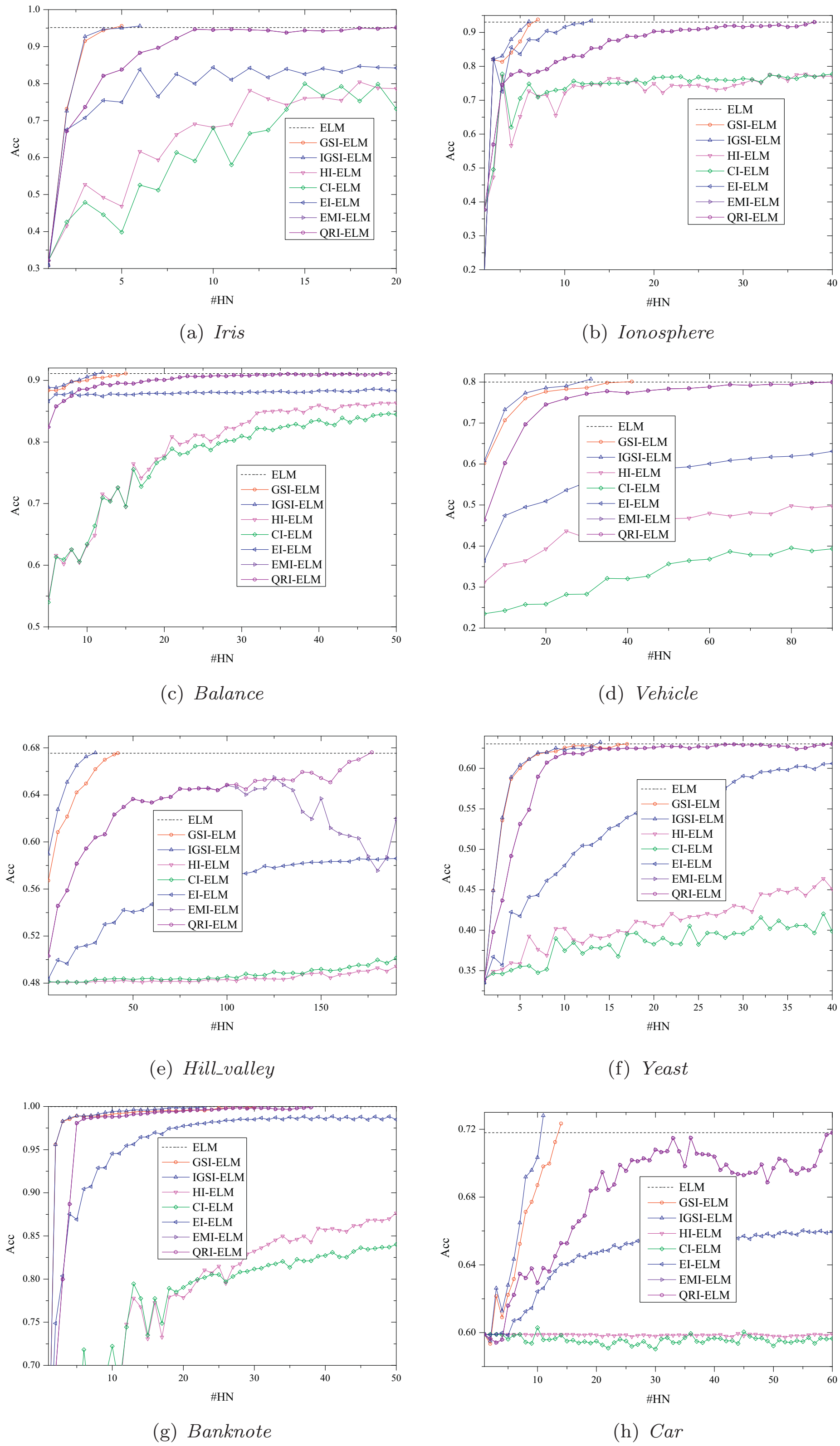
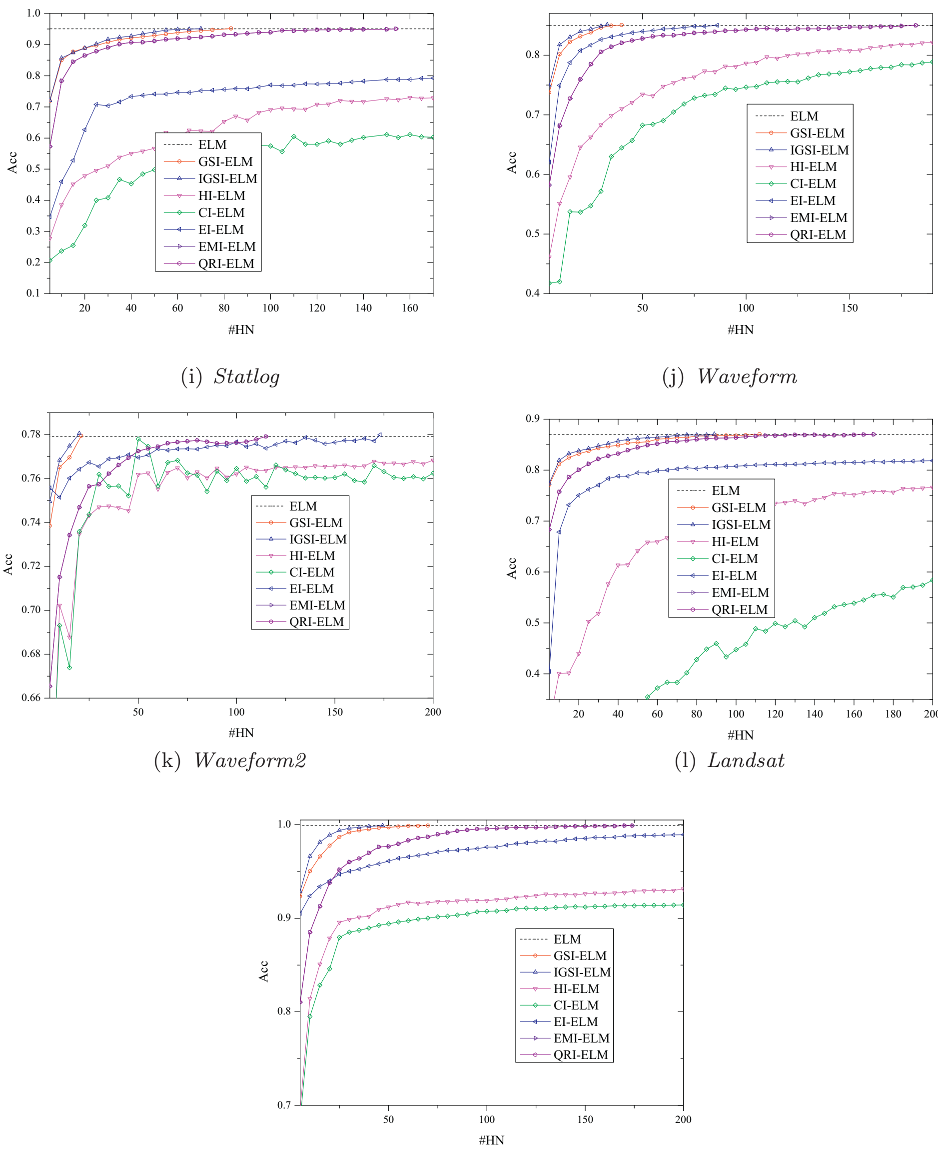


Fig. 2. Comparisons of the Acc in terms of #HN for different algorithms on classification applications.

signifies that IGSI-ELM is superior to GSI-ELM with respect to #HN under the same generalization performance. The main reason is due to the fact that IGSI-ELM adopts the elimination mechanism. As thus, the "nesting effect" can be overcome to a certain degree.

The experimental results are elaborated on in Table 2. From this table, these algorithms are terminated when they touch the dash line or they recruit the same #HN as the traditional ELM. In general, among HI-ELM, CI-ELM, and EI-ELM, the generalization performance of HI-ELM is worst, but EI-ELM owns the best gen-



(i) *Statlog*

(j) *Waveform*

(k) *Waveform2*

(l) *Landsat*

(m) *Mushroom*

Fig. 2. Continued

eralization performance. However, from the point of the training time, EI-ELM requires the longest while HI-ELM needs the shortest.

EMI-ELM and QRI-ELM usually need the less #HN when they obtain nearly the same generalization performance as the traditional ELM. Although EMI-ELM and QRI-ELM have the same RMSE vs. #HN, QRI-ELM obviously reduces the training time in comparison with EMI-ELM. As for this point, it is also proved by Ye and Qin [12]. When an evaluating criterion is introduced into QRI-ELM, the quality of the hidden nodes recruited is improved. Hence, GSI-ELM requires fewer hidden nodes than QRI-ELM under nearly the same level of the generalization performance, which results in those phenomena generated that GSI-ELM maybe needs less training time than QRI-ELM even though there is an extra evaluating criterion in GSI-ELM. Because of the elimination mechanism, IGSI-ELM works better than GSI-ELM in terms of #HN. That is, IGSI-ELM commonly needs fewer hidden nodes than GSI-ELM under nearly the same generalization performance, which means that the elimination mechanism is effective. Meanwhile, the “nesting effect” existing in GSI-ELM is mitigated. However, this elimination mechanism simultaneously brings extra training time to IGSI-ELM, thus leading to IGSI-ELM needing more training time than GSI-ELM. All in all, IGSI-ELM is the winner among all the competitors with respect to #SN. The fewest hidden nodes indicate the least testing time, because the testing time is directly proportional to #HN, which is especially suitable for the testing time sensitive scenarios.

6.2. Classification applications

Similar to regression applications, the prediction accuracy of the traditional ELM is chosen as the benchmark line (the dash line), as shown in Fig. 2. To facilitate comparison, when the prediction accuracy of the other algorithms arrives at the benchmark line or they require the same #HN as the traditional ELM, we terminate them.

From Fig. 2, it is observed that the prediction accuracy of HI-ELM and CI-ELM always do not reach the dash line when they recruit the same #HN as the traditional ELM. EI-ELM sometimes can touch the benchmark line, with only three out of thirteen. Among them, EI-ELM converges fastest.

Table 3
Performance comparison on classification applications

Data sets	Algorithms	L	Acc	Training time (s)	Testing time (s)
Iris	ELM	20	0.9515 ± 0.0163	0.002 ± 0.000	0.000 ± 0.000
	GSI-ELM	**5**	0.9561 ± 0.0106	0.002 ± 0.000	0.000 ± 0.000
	IGSI-ELM	6	0.9561 ± 0.0185	0.005 ± 0.001	0.000 ± 0.000
	HI-ELM	20	0.7864 ± 0.0652	**0.001 ± 0.000**	0.000 ± 0.000
	CI-ELM	20	0.7318 ± 0.1347	0.002 ± 0.000	0.000 ± 0.000
	EI-ELM	20	0.8424 ± 0.0182	0.018 ± 0.002	0.000 ± 0.000
	EMI-ELM	20	0.9515 ± 0.0163	0.004 ± 0.000	0.000 ± 0.000
	QRI-ELM	20	0.9515 ± 0.0163	0.002 ± 0.000	0.000 ± 0.000
Ionoshpere	ELM	40	0.9306 ± 0.0196	**0.002 ± 0.000**	0.000 ± 0.000
	GSI-ELM	7	0.9382 ± 0.0178	0.005 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**6**	0.9318 ± 0.0148	0.009 ± 0.002	0.000 ± 0.000
	HI-ELM	40	0.7707 ± 0.0856	**0.002 ± 0.000**	0.000 ± 0.000
	CI-ELM	40	0.7771 ± 0.0849	0.003 ± 0.000	0.000 ± 0.000
	EI-ELM	13	0.9350 ± 0.0110	0.014 ± 0.001	0.000 ± 0.000
	EMI-ELM	38	0.9306 ± 0.0220	0.018 ± 0.001	0.000 ± 0.000
	QRI-ELM	38	0.9306 ± 0.0220	0.005 ± 0.001	0.000 ± 0.000
Balance	ELM	50	0.9110 ± 0.0053	**0.002 ± 0.000**	0.000 ± 0.000
	GSI-ELM	15	0.9110 ± 0.0042	0.012 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**12**	0.9128 ± 0.0029	0.020 ± 0.004	0.000 ± 0.000
	HI-ELM	50	0.8633 ± 0.0162	**0.002 ± 0.000**	0.000 ± 0.000
	CI-ELM	50	0.8448 ± 0.0306	0.005 ± 0.001	0.000 ± 0.000
	EI-ELM	50	0.8833 ± 0.0076	0.056 ± 0.002	0.000 ± 0.000
	EMI-ELM	49	0.9110 ± 0.0039	0.051 ± 0.003	0.000 ± 0.000
	QRI-ELM	49	0.9110 ± 0.0039	0.007 ± 0.001	0.000 ± 0.000
Vehicle	ELM	90	0.8000 ± 0.0130	**0.003 ± 0.001**	0.001 ± 0.000
	GSI-ELM	41	0.8010 ± 0.0109	0.031 ± 0.002	0.000 ± 0.000
	IGSI-ELM	**31**	0.8071 ± 0.0227	0.067 ± 0.007	0.000 ± 0.000
	HI-ELM	90	0.4976 ± 0.0223	0.005 ± 0.001	0.001 ± 0.000
	CI-ELM	90	0.3934 ± 0.0483	0.009 ± 0.001	0.001 ± 0.000
	EI-ELM	90	0.6310 ± 0.0123	0.120 ± 0.002	0.001 ± 0.000
	EMI-ELM	90	0.8000 ± 0.0130	0.363 ± 0.004	0.001 ± 0.000
	QRI-ELM	90	0.8000 ± 0.0130	0.015 ± 0.000	0.001 ± 0.000
Hill_valley	ELM	190	0.6754 ± 0.0176	**0.013 ± 0.001**	0.003 ± 0.000
	GSI-ELM	42	0.6754 ± 0.0087	0.061 ± 0.001	0.001 ± 0.000
	IGSI-ELM	**30**	0.6758 ± 0.0121	0.121 ± 0.014	0.001 ± 0.000
	HI-ELM	190	0.4943 ± 0.0060	0.026 ± 0.001	0.003 ± 0.000
	CI-ELM	190	0.5015 ± 0.0128	0.029 ± 0.002	0.003 ± 0.000
	EI-ELM	190	0.5859 ± 0.0022	0.327 ± 0.003	0.003 ± 0.000
	EMI-ELM	190	0.6194 ± 0.0282	2.104 ± 0.010	0.003 ± 0.000
	QRI-ELM	177	0.6761 ± 0.0133	0.066 ± 0.001	0.003 ± 0.000
Yeast	ELM	40	0.6303 ± 0.0037	**0.002 ± 0.000**	0.001 ± 0.000
	GSI-ELM	17	0.6301 ± 0.0057	0.020 ± 0.001	0.000 ± 0.000
	IGSI-ELM	**14**	0.6324 ± 0.0096	0.029 ± 0.003	0.000 ± 0.000
	HI-ELM	40	0.4508 ± 0.0253	0.003 ± 0.000	0.001 ± 0.000
	CI-ELM	40	0.3986 ± 0.0240	0.005 ± 0.000	0.001 ± 0.000
	EI-ELM	40	0.6060 ± 0.0121	0.073 ± 0.002	0.001 ± 0.000
	EMI-ELM	40	0.6303 ± 0.0037	0.339 ± 0.003	0.001 ± 0.000
	QRI-ELM	40	0.6303 ± 0.0037	0.007 ± 0.001	0.001 ± 0.000
Banknote	ELM	50	0.9998 ± 0.0005	**0.003 ± 0.000**	0.001 ± 0.000
	GSI-ELM	30	0.9993 ± 0.0015	0.036 ± 0.002	0.000 ± 0.000
	IGSI-ELM	**23**	0.9993 ± 0.0015	0.072 ± 0.009	0.000 ± 0.000
	HI-ELM	50	0.8764 ± 0.0325	**0.003 ± 0.000**	0.001 ± 0.000
	CI-ELM	50	0.8402 ± 0.0431	0.004 ± 0.000	0.001 ± 0.000
	EI-ELM	50	0.9848 ± 0.0016	0.076 ± 0.002	0.001 ± 0.000
	EMI-ELM	38	0.9990 ± 0.0017	0.366 ± 0.003	0.001 ± 0.000
	QRI-ELM	38	0.9990 ± 0.0017	0.007 ± 0.001	0.001 ± 0.000
Car	ELM	60	0.7180 ± 0.0441	**0.003 ± 0.001**	0.001 ± 0.000
	GSI-ELM	14	0.7234 ± 0.0193	0.021 ± 0.002	0.000 ± 0.000
	IGSI-ELM	**11**	0.7280 ± 0.0186	0.029 ± 0.006	0.000 ± 0.000
	HI-ELM	60	0.5992 ± 0.0053	0.005 ± 0.000	0.001 ± 0.000
	CI-ELM	60	0.5965 ± 0.0210	0.009 ± 0.000	0.001 ± 0.000
	EI-ELM	60	0.6594 ± 0.0055	0.128 ± 0.001	0.001 ± 0.000
	EMI-ELM	60	0.7180 ± 0.0441	0.982 ± 0.014	0.001 ± 0.000
	QRI-ELM	60	0.7180 ± 0.0441	0.013 ± 0.000	0.001 ± 0.000
Statlog	ELM	170	0.9509 ± 0.0046	**0.013 ± 0.001**	0.005 ± 0.000
	GSI-ELM	83	0.9517 ± 0.0041	0.155 ± 0.009	0.002 ± 0.000
	IGSI-ELM	**70**	0.9514 ± 0.0036	0.571 ± 0.053	0.002 ± 0.000
	HI-ELM	170	0.7291 ± 0.0066	0.017 ± 0.001	0.005 ± 0.000
	CI-ELM	170	0.6024 ± 0.0418	0.034 ± 0.001	0.005 ± 0.000

(*continued on next page*)

Table 3 (*continued*)

Data sets	Algorithms	*L*	Acc	Training time (s)	Testing time (s)
	EI-ELM	170	0.7925 ± 0.0082	0.550 ± 0.005	0.005 ± 0.000
	EMI-ELM	154	0.9510 ± 0.0042	4.347 ± 0.015	0.004 ± 0.000
	QRI-ELM	154	0.9510 ± 0.0042	0.095 ± 0.001	0.004 ± 0.000
Waveform	ELM	190	0.8500 ± 0.0038	**0.036 ± 0.003**	0.011 ± 0.000
	GSI-ELM	40	0.8505 ± 0.0043	0.222 ± 0.004	0.003 ± 0.000
	IGSI-ELM	**33**	0.8512 ± 0.0072	0.493 ± 0.054	0.003 ± 0.000
	HI-ELM	190	0.8217 ± 0.0136	0.046 ± 0.001	0.011 ± 0.000
	CI-ELM	190	0.7887 ± 0.0163	0.058 ± 0.002	0.011 ± 0.001
	EI-ELM	86	0.8500 ± 0.0035	0.557 ± 0.008	0.006 ± 0.000
	EMI-ELM	182	0.8500 ± 0.0063	28.958 ± 0.106	0.010 ± 0.001
	QRI-ELM	182	0.8500 ± 0.0063	0.437 ± 0.002	0.010 ± 0.001
Waveform2	ELM	200	0.7791 ± 0.0046	0.060 ± 0.027	0.012 ± 0.000
	GSI-ELM	21	0.7795 ± 0.0030	0.116 ± 0.003	0.002 ± 0.000
	IGSI-ELM	**20**	0.7806 ± 0.0070	0.265 ± 0.033	0.002 ± 0.000
	HI-ELM	200	0.7684 ± 0.0056	**0.047 ± 0.003**	0.012 ± 0.000
	CI-ELM	200	0.7625 ± 0.0083	0.056 ± 0.002	0.012 ± 0.000
	EI-ELM	173	0.7800 ± 0.0035	0.985 ± 0.005	0.010 ± 0.000
	EMI-ELM	115	0.7792 ± 0.0094	16.146 ± 0.051	0.008 ± 0.000
	QRI-ELM	115	0.7792 ± 0.0094	0.175 ± 0.003	0.008 ± 0.000
Landsat	ELM	200	0.8704 ± 0.0021	**0.053 ± 0.002**	0.015 ± 0.000
	GSI-ELM	112	0.8706 ± 0.0033	0.960 ± 0.005	0.007 ± 0.000
	IGSI-ELM	**89**	0.8704 ± 0.0027	5.208 ± 0.624	0.007 ± 0.000
	HI-ELM	200	0.7663 ± 0.0143	0.125 ± 0.002	0.015 ± 0.000
	CI-ELM	200	0.5838 ± 0.0771	0.168 ± 0.002	0.015 ± 0.000
	EI-ELM	200	0.8185 ± 0.0019	2.040 ± 0.011	0.015 ± 0.000
	EMI-ELM	170	0.8703 ± 0.0018	43.104 ± 0.135	0.012 ± 0.000
	QRI-ELM	170	0.8703 ± 0.0018	0.542 ± 0.003	0.012 ± 0.000
Mushroom	ELM	200	0.9994 ± 0.0007	0.070 ± 0.007	0.016 ± 0.001
	GSI-ELM	70	0.9990 ± 0.0008	0.587 ± 0.012	0.007 ± 0.000
	IGSI-ELM	**47**	0.9990 ± 0.0011	1.290 ± 0.149	0.005 ± 0.000
	HI-ELM	200	0.9314 ± 0.0159	**0.046 ± 0.002**	0.016 ± 0.001
	CI-ELM	200	0.9142 ± 0.0107	0.061 ± 0.001	0.016 ± 0.001
	EI-ELM	200	0.9893 ± 0.0025	1.679 ± 0.009	0.016 ± 0.001
	EMI-ELM	174	0.9990 ± 0.0011	53.814 ± 0.146	0.012 ± 0.001
	QRI-ELM	174	0.9990 ± 0.0011	0.565 ± 0.002	0.012 ± 0.001

EMI-ELM and QRI-ELM own the same prediction accuracy under the same #HN except the *Hill_valley* case. In the case of *Hill_valley*, the generalization performance of EMI-ELM deteriorates because of the round off errors, which demonstrates that QRI-ELM is more stable than EMI-ELM.

GSI-ELM boosts the prediction accuracy by improving the quality of the hidden nodes recruited compared with QRI-ELM/EMI-ELM, which shows that the evaluating criterion defined in this paper is effective and the probabilistic trick utilized is feasible. Though adding the elimination mechanism, IGSI-ELM enhances the generalization performance further, which means that IGSI-ELM needs fewer hidden nodes than GSI-ELM under nearly the same generalization performance.

Table 3 showcases the detailed experimental results on classification applications. When keeping the same number of hidden nodes as the traditional ELM, EI-ELM owns the best generalization performance among HI-ELM, CI-ELM, and EI-ELM, but HI-ELM performs best with respect to the training time. Although EMI-ELM and QRI-ELM have the same Acc, EMI-ELM is inferior to QRI-ELM in terms of the training time. In contrast with QRI-ELM, GSI-ELM reduces #HN obviously and retains the comparative training time under nearly the same generalization performance. IGSI-ELM outperforms GSI-ELM in #HN but loses the advantage in the training time by exploiting the elimination mechanism. In a word, those above conclusions obtained on classification applications are nearly consistent with those on regression applications.

7. Conclusions

In recent years, extreme learning machine has become a popular topic in the machine learning community. Different from the traditional SLFNs, ELM chooses the input weights and the hidden layer biases randomly, which maybe produces the negative effect that the negligible hidden nodes, which play a very minor role in the network output, may be recruited. As a result, its architecture is not compact, which is not suitable for the testing time sensitive scenarios. To obtain a more compact architecture, the incremental learning algorithms are developed. Hence, two incremental learning algorithms, viz. GSI-ELM and IGSI-ELM, are proposed based on QRI-ELM in this paper. In QRI-ELM, the hidden nodes are incrementally recruited, which is equivalent to randomly selecting a hidden node from a candidate set of infinite size at each learning step. That is to say, the problem of recruiting the negligible hidden nodes to construct the hidden layer of ELM still exists. It is impossible to recruit the *best* hidden node from the candidate set of infinite size. To improve the quality of the hidden nodes recruited in QRI-ELM, an evaluating criterion is defined firstly, and then a probabilistic trick is utilized to recruit a *best* hidden node from a random subset of fixed size at each learning step. This feasible strategy assists GSI-ELM in gaining better performance with respect to #HN. However, the “nesting effect” exists in GSI-ELM. To treat this “nesting problem”, an elimination mechanism is added to GSI-ELM. To implement this elimination mechanism, an evaluation criterion δ_i is defined on the existing hidden nodes firstly. Then, an accelerating scheme of δ_i is presented to cut down the computational complexity. Due to the addition of the elimination mechanism, IGSI-ELM overcomes the “nesting effect” to a certain degree, thus yielding better performance in #HN, but requiring longer training time. Through comparing with the other incremental learning algorithms, the proposed GSI-ELM and IGSI-ELM in this paper are experimentally favored.

Acknowledgments

This research was partially supported by the Fundamental Research Funds for the Central Universities under Grant no. NJ20160021, and the National Natural Science Foundation of China under Grand no. 11502008.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 2, 2004, pp. 985–990. Budapest, Hungary, [Online]. Available: http://dx.doi.org/10.1109/IJCNN.2004.1380068

[2] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2005.12.126

[3] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, doi:10.1038/323533a0.

[4] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423. [Online]. Available: http://dx.doi.org/10.1109/TNN.2006.880583

[5] H.T. Huynh, Y. Won, Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks, Pattern Recognit. Lett. 32 (14) (2011) 1930–1935. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2011.07.016

[6] Z. Shao, M.J. Er, An online sequential learning algorithm for regularized extreme learning machine, Neurocomputing 173 (2016) 778–788. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2015.08.029

[7] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892. [Online]. Available: http://dx.doi.org/10.1109/TNN.2006.875977

[8] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2007.02.009

[9] G.-B. Huang, L. Chen, Enhanced random search based incremental extreme learning machine, Neurocomputing 71 (16–18) (2008) 3460–3468. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2007.10.008

[10] G. Feng, G.-B. Huang, Q. Lin, R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Trans. Neural Netw. 20 (8) (2009) 1352–1357. [Online]. Available: http://dx.doi.org/10.1109/TNN.2009.2024147

[11] R. Zhang, Y. Lan, G.-B. Huang, Z.-B. Xu, Universal approximation of extreme learning machine with adaptive growth of hidden nodes, IEEE Trans. Neural Netw. Learn. Syst. 23 (2) (2012) 365–371. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2011.2178124

[12] Y. Ye, Y. Qin, Qr factorization based incremental extreme learning machine with growth of hidden nodes, Pattern Recognit. Lett. 65 (2015) 177–183. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2015.07.031

[13] Y.-P. Zhao, K.-K. Wang, Y.-B. Li, Parsimonious regularized extreme learning machine based on orthogonal transformation, Neurocomputing 156 (2015) 280–296. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2014.12.046

[14] Y.-P. Zhao, R. Huerta, Improvements on parsimonious extreme learning machine using recursive orthogonal least squares, Neurocomputing 191 (2016) 82–94. http://dx.doi.org/10.1016/j.neucom.2016.01.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231216000527

[15] H.-J. Rong, Y.-S. Ong, A.-H. Tan, Z. Zhu, A fast pruned-extreme learning machine for classification problem, Neurocomputing 72 (1–3) (2008) 359–366. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2008.01.005

[16] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, Op-elm: Optimally pruned extreme learning machine, IEEE Transactions on Neural Networks 21 (1) (2010) 158–162. [Online]. Available: http://dx.doi.org/10.1109/TNN.2009.2036259

[17] A. Castano, F. Fernandez-Navarro, C. Hervas-Martinez, Pca-elm: A robust and pruned extreme learning machine approach based on principal component analysis, Neural Process. Lett. 37 (3) (2013) 377–392. [Online]. Available: http://dx.doi.org/10.1007/s11063-012-9253-x

[18] Y.-P. Zhao, B. Li, Y.-B. Li, An accelerating scheme for destructive parsimonious extreme learning machine, Neurocomputing 167 (2015) 671–687. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2015.04.002

[19] A.S. Alencar, A.R. Rocha Neto, J.P.P. Gomes, A new pruning method for extreme learning machines via genetic algorithms, Applied Soft Computing vol. 44 (2016) 101–107. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2016.03.019

[20] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (2) (1991) 213–225.

[21] V. Kadirkamanathan, M. Niranjan, A function estimation approach to sequential learning with neural networks, Neural Comput. 5 (6) (1993). 954–954

[22] Y. Lu, N. Sundararajan, P. Saratchandran, Sequential learning scheme for function approximation using minimal radial basis function neural networks, Neural Comput. 9 (2) (1997). 461–461

[23] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inf. Theory 39 (3) (1993) 930–945.

[24] A. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: Proceedings of the International Conference on Machine Learning, 2000.

[25] B. Schölkopf, A.J. Smola, Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, 2002.

[26] Y.-P. Zhao, Parsimonious kernel extreme learning machine in primal via cholesky factorization, Neural Netw. 80 (2016) 95–109. http://dx.doi.org/10.1016/j.neunet.2016.04.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608016300399

[27] P. Pudil, J. Novovicová, J. Kittler, Floating search methods in feature selection, Pattern Recognit. Lett. 15 (11) (1994) 1119–1125. http://dx.doi.org/10.1016/0167-8655(94)90127-9.

[28] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, (second ed.), Wiley-Interscience, 2000.

[29] X. Zhang, Matrix Analysis and Applications, Tsinghua university press, 2004.

[30] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, 1996.

[31] Y. Zhao, K. Wang, Fast cross validation for regularized extreme learning machine, J. Syst. Eng. Electr. 25 (5) (2014) 895–900. [Online]. Available: http://dx.doi.org/10.1109/JSEE.2014.000103

Yong-Ping Zhao received his B.E. degree in the thermal energy and power engineering field from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in July 2004. Since then, he had been pursuing the M.S. and Ph.D. degrees at Nanjing University of Aeronautics and Astronautics. In December 2009, He received Ph.D. degree, and won the award of the Nominated for the National Excellent Doctoral Dissertation Award of China in 2013. Currently, he is a professor and with the college of energy and power engineering, Nanjing University of Aeronautics and Astronautics. His research interests include aircraft engine modeling, control and fault diagnostics, machine learning and pattern recognition.

Zhi-Qiang Li was born in 1993. He received the B.S. degree from the Jiangnan University in 2016. And he is currently pursuing the M.S degree form Nanjing University of Aeronautics and Astronautics (UNAA). His main research interests include feature selection of areo-engine, evolutionary computation, and support vector machine.

Peng-Peng Xi was born in Jiangxi, China, in 1994. He received the B.S. degrees from the Nanjing University of Aeronautics and Astronautics, in 2016. He is currently pursuing the M.S. degree from Nanjing University of Aeronautics and Astronautics. His research interest is aircraft engine modeling, control and fault diagnostics.

Dong Liang received the B.S. degree in measurement and control engineering from Nanchang Hangkong University, Nanchang, China, in 2005. After seven years of graduate study, he received the M.S. degree in aircraft engineering and the Ph.D. degree in measuring and testing technologies and instruments from Nanjing University of Aeronautics and Astronautics, Nanjing, China. He is an Assistant Professor at Department of Aeronautics, College of Physics and Electromechanics, Xiamen University, China. His research interests include distributed sensor network, compressive sensing and pattern recognition in aircraft structural health monitoring.

Liguo Sun received his B.Sc. and M.Sc. degrees from Nanjing University of Aeronautics and Astronautics (China) in 2008 and 2010. On October 30th of 2014, he received his Ph.D. degree from the Control and Simulation Group, Faculty of Aerospace Engineering, TU Delft. Since June of 2015, he has been an associate professor at the Flight Dynamics and Flight Safety group at the School of Aeronautic Science and Engineering, Beihang University. His current research interests include nonlinear system identification, machine learning, multivariate spline theory, fault-tolerant (nonlinear) flight control, aircraft safe-flight-envelope prediction, propulsion control.

Ting-Hao Chen received his B.S. degree in thermal energy and power engineering field from Civil Aviation Flight University of China, Guanghan, China, in 2007. He received M.S. degree in Aero-engine fault diagnosis field from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010. His research interests include Aero-engine fault diagnosis, machine learning, etc.