**Original Article**
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 16, No. 2, June 2016, pp. 125-130
http://dx.doi.org/10.5391/IJFIS.2016.16.2.125

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

# Pseudoinverse Matrix Decomposition Based Incremental Extreme Learning Machine with Growth of Hidden Nodes

**Peyman Hosseinzadeh Kassani and Euntai Kim**
School of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea

ljfis

**Abstract**

The proposal of this study is a fast version of the conventional extreme learning machine (ELM), called pseudoinverse matrix decomposition based incremental ELM (PDI-ELM). One of the main problems in ELM is to determine the number of hidden nodes. In this study, the number of hidden nodes is automatically determined. The proposed model is an incremental version of ELM which adds neurons with the goal of minimization the error of the ELM network. To speed up the model the information of pseudoinverse from previous step is taken into account in the current iteration. To show the ability of the PDI-ELM, it is applied to few benchmark classification datasets in the University of California Irvine (UCI) repository. Compared to ELM learner and two other versions of incremental ELM, the proposed PDI-ELM is faster.

**Keywords:** ELM, Pseudoinverse matrix, Hidden nodes, Least square estimation, Speed up

## 1. Introduction

Recently, extreme learning machine (ELM) has attracted much attention in the area of machine learning [1–5]. Due to closed form solution, and free from learning input weights, ELM is among popular learners. The structure of ELM backs to the single hidden layer feedforward networks (SLFNs). Huang et al. [6] theoretically proved that the hidden nodes in SLFNs can be randomly generated and be relaxed from learning. This explicit feature mapping plus analytically determination of the output weights using least square method leads to a huge reduction in the training time. There are many variants of ELM with different goals [7–9]. A main drawback of ELM is as the number of hidden nodes grows, the ELM networks more likely prone to the overfitting problem [10]. Several researches are conducted to resolve this problem. On one hand, we wish to increase the number of hidden nodes to recognize the complex relationships among data, and on the other hand, the model should avoid overfitting problem to get good performance on test data.

The number of hidden nodes in ELM should be assigned by the user which is done by trial and error. As the size of data increases, more hidden nodes are required to get better accuracy. However it is still unknown how many neurons are enough as the size of data grows. There should be an exhaustive work using cross validation technique or other ways to find the exact size of hidden nodes.

The first study which incrementally adds hidden nodes to reach a pre-determined network error is referred to as incremental ELM (I-ELM) [10]. This model adds nodes one by one and

Received: Jun. 10, 2016
Revised : Jun. 16, 2016
Accepted: Jun. 16, 2016

Correspondence to: Euntai Kim
(etkim@yonsei.ac.kr)

freezes the output weights of the current network when a new node is added.

In this study we try to make a new competitive I-ELM network in terms of pseudoinverse matrix decomposition. The proposed model is abbreviated to PDI-ELM. This method is competitive with regard to the speed of the algorithm and can compete with its newest versions of I-ELMs which are briefly introduced in next section.

Section 2 describes the related work and Section 3 outlines the ELM and all the stages of the proposed PDI-ELM learner. Section 4 shows results of ELM, EM-ELM, QRI-ELM and the proposed PDI-ELM on four portions of the University of California Irvine (UCI) data repository to illustrate and compare the performance of models. Conclusions are drawn briefly in Section 5.

## 2. Related Work

### 2.1 Extreme Learning Machine

Let assume $N$ is the number of training samples and $d$ is the dimension of feature size from dataset $\mathbf{D} = (\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, ..., N$ where $\mathbf{x}_i$ is input vector and $\mathbf{t}_i$ is desirable output. For ELM neural network with $L$ hidden neuron, the network output is [11]:

$$f(\mathbf{x}) = \sum_{k=0}^{L} w_k h_k(\theta_k; \mathbf{x}) = \mathbf{h}(\Theta; \mathbf{x})\mathbf{w}, \tag{1}$$

where $\mathbf{h}(\Theta; \mathbf{x}) = [1, h_1(\theta_1; \mathbf{x}), ..., h_L(\theta_L; \mathbf{x})]$ maps the feature in hidden layer by given input $\mathbf{x}$. $\Theta = [\theta_1, ..., \theta_L]$ are random parameters come from a uniform distribution, and $\mathbf{w}$ is the weight vector of all neurons in hidden layer to the neuron in output layer and finally $h_k(.)$ is the activation function of hidden layer. In a compact form, Eq. (1) can be written as:

$$\mathbf{HB} = \mathbf{T}, \tag{2}$$

where $\mathbf{H}$ is the kernel matrix and its elements plus bias term is as follows:

$$\mathbf{H} = \begin{bmatrix} 1 & h_1(\theta_1; \mathbf{x}_1) & . & h_L(\theta_L; \mathbf{x}_1) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & h_1(\theta_1; \mathbf{x}_N) & . & h_L(\theta_L; \mathbf{x}_N) \end{bmatrix}. \tag{3}$$

Each row of $\mathbf{H}$ is the output vector of a sample $\mathbf{x}$. Given all elements of matrix $\mathbf{H}$ , the goal is finding optimal weight vector $\mathbf{B}$ (optimality in this sense that the desirable target matrix $\mathbf{T}$ can be estimated well) which is:

$$\mathbf{B} = \mathbf{H}^{\dagger}\mathbf{T} = \left(\mathbf{H^T H}\right)^{-1}\mathbf{H^T T}. \tag{4}$$

Estimated $\mathbf{B}$ is substituted into (2) to obtain the target matrix $\mathbf{T}$.

### 2.2 Incremental Based Extreme Learning Machines

A new algorithm of the I-ELM, called error minimized ELM (EM-ELM) is proposed at [12]. This network can add nodes one by one or group by group (Chunking). In every step, EM-ELM minimizes error. This networks gets a better generalization performance than I-ELM and works faster than I-ELM. The key points of all I-ELM methods is as a new hidden node is added to the network we expect to get smaller error of network than before addition. This can be mathematically seen as follows:

**Lemma 1** [12]. *Given an SLFN, let $\mathbf{H_1}$ be the initial hidden layer output matrix with $L_0$ hidden nodes. If $L_1 - L_0$ hidden nodes are added to the current network to make the new hidden layer output matrix (Let call $\mathbf{H_2}$), then we have $E(H_2) < E(H_1)$.*

The simple proof of this lemma can be found at [12].

In EM-ELM model, $\mathbf{H}^{\dagger}_{k+1}$ is iteratively found via $\mathbf{H}^{\dagger}_{k}$. When one neuron is added to the existing network with $k$ hidden nodes, a new column called $\mathbf{h}_{k+1}$ is added to $\mathbf{H}_k$ and the $\mathbf{H}_{k+1}$ is formed. To speed up this model, the $\mathbf{H}^{\dagger}_{k+1}$ is decomposed into two parts; $\mathbf{U}$ and $\mathbf{D}$. Then, using Schur complement and matrix block inversion lemma, $\mathbf{U}$ and $\mathbf{D}$ are found like this:

$$\mathbf{D}_{k+1} = \frac{\mathbf{h}^T_{k+1} - \mathbf{h}^T_{k+1}\mathbf{H}_k\mathbf{H}^{\dagger}_k}{\mathbf{h}^T_{k+1}\mathbf{h}_{k+1} - \mathbf{h}^T_{k+1}\mathbf{H}_k\mathbf{H}^{\dagger}_k\mathbf{h}_{k+1}} \tag{5}$$

$$\mathbf{U}_{k+1} = \mathbf{H}^{\dagger}_k - \mathbf{H}^{\dagger}_k\mathbf{h}_{k+1}\mathbf{D}_k. \tag{6}$$

And finally:

$$\mathbf{B}_{k+1} = \begin{bmatrix} \mathbf{U}_{k+1} \\ \mathbf{D}_{k+1} \end{bmatrix} \mathbf{T}. \tag{7}$$

From computationally aspect, this way to find output weights $\mathbf{B}$ takes lower time than traditional ELM model.

The latest version of I-ELM based methods, called QRI-ELM [13] decomposes the pseudoinverse matrix of the hidden output layer based on QR factorization. Indeed, **H=Q.R** wherein $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. Hence, $\mathbf{H}^{\dagger} = \mathbf{R}^{-1}.\mathbf{Q}^T$. This way simplifies finding the pseudoinverse of **H**. QRI-ELM gets similar performance to EM-ELM and I-ELM but is faster than other variants.

## 3. Proposed Method

In this section, our proposed method which is PDI-ELM algorithm is introduced. Let us consider **H** and its pseudoinverse in the partitioned form [14]:

$$\mathbf{H}_k = \begin{pmatrix} \mathbf{H}_{k-1} & \mathbf{h}_k \end{pmatrix} \quad , \quad \mathbf{H}_k^{\dagger} = \begin{pmatrix} \mathbf{G}_k \\ \mathbf{g}_k \end{pmatrix}. \tag{8}$$

Their multiplication gives:

$$\mathbf{H}_k\mathbf{H}_k^{\dagger} = \mathbf{H}_{k-1}\,\mathbf{G}_k \;+\; \mathbf{h}_k\mathbf{g}_k. \tag{9}$$

Now, if we further multiply above from left hand in $\mathbf{H}_{k+1}^{\dagger}$ :

$$\mathbf{H}_{k-1}^{\dagger} = \mathbf{G}_k + \mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k\mathbf{g}_k. \tag{10}$$

The above is attained because the following equations are hold:

$$\mathbf{H}_{k-1}^{\dagger}\mathbf{H}_k\mathbf{H}_k^{\dagger} = \mathbf{H}_{k-1}^{\dagger} \quad , \quad \mathbf{H}_{k-1}^{\dagger}\mathbf{H}_{k-1}\mathbf{G}_k = \mathbf{G}_k. \tag{11}$$

Therefore, the pseudoinverse matrix changes to:

$$\mathbf{H}_k^{\dagger} = \begin{pmatrix} \mathbf{H}_{k-1}^{\dagger} - \mathbf{d}_k\mathbf{g}_k \\ \mathbf{g}_k \end{pmatrix}, \tag{12}$$

where:

$$\mathbf{d}_k = \mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k. \tag{13}$$

Now the goal is converted to finding $\mathbf{g}_k$. The product of $\mathbf{H}_k$ in (8) and $\mathbf{H}_k^{\dagger}$ in (12) gives:

$$\mathbf{H}_k\mathbf{H}_k^{\dagger} = \;\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{g}_k, \tag{14}$$

where:

$$\mathbf{c}_k = \mathbf{h}_k - \mathbf{H}_{k-1}\mathbf{d}_k. \tag{15}$$

And multiplying (12) from left hand by $\mathbf{H}_{k-1}^{\dagger}$ with considering (10) gives:

$$\mathbf{H}_{k-1}^{\dagger}\mathbf{c}_k = \mathbf{0}. \tag{16}$$

The above equation means that $\mathbf{c}_k$ is orthogonal to the column-space of $\mathbf{H}_{k-1}$. If the N training data are distinct, **H** is full column rank with probability one when the number of hidden nodes is equal or less than the number of samples *N* [11]. So, we can conclude that $\mathbf{c}_k$ is not equal to zero. Hence:

$$\mathbf{c}_k^{\dagger} \;\times\; (\mathbf{c}_k = \mathbf{h}_k - \mathbf{H}_{k-1}\mathbf{d}_k) \quad\rightarrow\quad \mathbf{c}_k^{\dagger}\mathbf{h}_k = 1. \tag{17}$$

The above is attained because $\mathbf{c}_k^{\dagger}\mathbf{c}_k = 1$ and we know that $\left(\mathbf{H}_{k-1}^{\dagger}\mathbf{c}_k\right)^{\dagger} = \mathbf{c}_k^{+}\mathbf{H}_{k-1} = 0$. Relying on (14) we would like to know whether $\mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{g}_k \;=\; \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{c}_k^{\dagger}$ or not. Let us assume:

$$\mathbf{P}_k = \mathbf{H}_{k-1}\mathbf{H}_{k-1}^{\dagger} + \mathbf{c}_k\mathbf{c}_k^{\dagger}. \tag{18}$$

It is not hard to see that:

$$\mathbf{P}_k\mathbf{h}_k = \mathbf{h}_k \;,\; \mathbf{P}_k\mathbf{H}_k = \mathbf{H}_k \;\rightarrow\; \mathbf{P}_k \;=\; \mathbf{H}_k\mathbf{H}_k^{\dagger}. \tag{19}$$

This leads to find:

$$\mathbf{c}_k\mathbf{c}_k^{\dagger} = \mathbf{c}_k\mathbf{g}_k \qquad\rightarrow\qquad \mathbf{g}_k = \mathbf{c}_k^{\dagger}. \tag{20}$$

Putting (13), (15), (20) into (12) leads to finding $\mathbf{H}_k^{\dagger}$. Now we summarize the whole steps of PDI-ELM algorithm as follows:

**Algorithm 1:** PDI-ELM Algorithm

```
. Given a set of training data, the maximum number of
  Hidden nodes $L_{max}$ :
. Randomly generate the first hidden node $L_0$.
. Calculate the hidden layer output matrix $\mathbf{h_1}$.
. Randomly generate the first hidden node $L_0$.
. Compute the Output error E($\mathbf{h_1}$).
// Recursive incremental Hidden nodes
. While $L_k < L_{max}$
    . k = k+1.
    . Randomly generate new hidden layer column and add
      the hidden node to the current SLFN.
    . $\mathbf{d}_k = \mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k$
    . $\mathbf{c}_k = \mathbf{h}_k - \mathbf{H}_{k-1}\mathbf{d}_k$
    . $\mathbf{g}_k = \mathbf{c}_k^{\dagger}$
    . $\mathbf{H}_k^{\dagger} = \begin{pmatrix} \mathbf{H}_{k-1}^{\dagger} - \mathbf{d}_k\mathbf{g}_k \\ \mathbf{g}_k \end{pmatrix}$
    . $\mathbf{B}_k = \mathbf{H}_k^{\dagger}\mathbf{T}$
. End While.
```

Note that If **A** is a matrix with m rows and n columns (mn) and **B** has n rows and c columns, then, there will be mnc element multiplications in their matrix product. So computational complexity for **AB** is O(mnc). The most computational time is for $\mathbf{g}_k = \mathbf{c}_k^{\dagger}$. Because it uses singular value decomposition to compute the pseudoinverse of $\mathbf{c}_k$ which is computationally expensive. Moreover, we could change stopping condition

Table 1. Descriptions of datasets

| Dataset | Train | Test | Attributes (R/I/N)* |
|---|---|---|---|
| WBCD | 630 | 69 | 9 (0/9/0) |
| Ionosphere | 310 | 41 | 33 (32/1/0) |
| Pima | 576 | 192 | 8 (8/0/0) |
| Abalone | 3500 | 677 | 8 (8/0/0) |

R, real; I, integer; N, normal.

in the while loop to be a predefined error of network (E) instead to reach the maximum number of hidden nodes. It is worthwhile to point out that for $\mathbf{d}_k = \mathbf{H}_{k-1}^{\dagger}\mathbf{h}_k$ only for the first hidden node we have to compute the pseudoinverse of $\mathbf{H}$ and for next iterations there is no need to compute it since $\mathbf{H}_k^{\dagger} = \begin{pmatrix} \mathbf{H}_{k-1}^{\dagger} - \mathbf{d}_k\mathbf{g}_k \\ \mathbf{g}_k \end{pmatrix}$ is found via $\mathbf{H}_{k-1}^{\dagger}$. So, this is the key point why we use pseudoinverse decomposition and that is why our proposed model works faster than ELM.

## 4. Experiments

### 4.1 Dataset Description

To verify the reliability of the proposed model, four benchmark datasets of the UCI data repository are adopted, and results are compared with some state-of-the-art classifiers. The datasets are listed in the Table 1.

#### 4.1.1 Experimental design

The sigmoid function is used for the hidden layer activation function of the ELM. Input weights and biases are chosen from a uniform distribution within [1, 1]. In addition, to evaluate ELM performance, each algorithm is 50 times run and the average results are reported. The algorithms are implemented on a personal computer with an Intel processor, i5 core, 3.4 GHz, and 8 GB installed RAM.

#### 4.1.2 Experiment results

This section gives an evaluation of the proposed method for PDI-ELM model.

Table 2 reveals that PDI-ELM has slightly better train and test time than QRI-ELM and better than EM-ELM model while all models have very similar performance on test samples.

Although the results indicate the slightly better speed up for our proposed PDI-ELM model, it is good to examine the computational complexity in terms of big O. Hence, Table 3 shows the computational complexity of ELM, EM-ELM, QRI-ELM and PDI-ELM. The details regarding how to find computational complexities for three compared models can be found at [13].

Table 2. Performance comparison in benchmark datasets (when certain numbers of hidden nodes are reached)

| Dataset | Model | Nodes | TR time | TE time | TE Acc |
|---|---|---|---|---|---|
| WBCD | EM | 150 | 0.0033 | 7.50e-5 | 0.9648 |
| | QRI | | 0.0029 | 4.17e-5 | 0.9642 |
| | PDI | | 0.0027 | 4.03e-5 | 0.9642 |
| Iono | EM | 100 | 0.0027 | 3.08e-5 | 0.8803 |
| | QRI | | 0.0022 | 2.86e-5 | 0.8811 |
| | PDI | | 0.0020 | 2.49e-5 | 0.8815 |
| Pima | EM | 150 | 0.0025 | 2.11e-4 | 0.7755 |
| | QRI | | 0.0018 | 1.57e-4 | 0.7768 |
| | PDI | | 0.0015 | 1.34e-4 | 0.7745 |
| Abalone | EM | 300 | 0.0972 | 6.45e-3 | 0.6578 |
| | QRI | | 0.0255 | 3.20e-3 | 0.6499 |
| | PDI | | 0.0184 | 1.45e-3 | 0.6506 |

Table 3. Computational complexities of a few incremental ELM algorithms

| Algorithm | Computational complexity |
|---|---|
| PDI-ELM | O($(4L+2)N$) |
| QRI-ELM [13] | O($(2L+3)N$) |
| EM-ELM [12] | O($(5L+2)N$) |
| ELM [11] | O($2L^2N+L^3$) |

Our proposed PDI-ELM is the first best one and QRI-ELM is the second best algorithm. QRI-ELM is slightly slower than PDI-ELM. It is worth noting that usually the number of training samples $N$ is much bigger than the number of hidden nodes $L$.

Figure 1 shows the number of hidden nodes one time versus CPU time and another time versus mean square error for two Pima and Ionosphere datasets. According to the plots, our proposed PDI-ELM is slightly better than QRI-ELM in terms of CPU time with similar performance in terms of MSE. PDI-ELM is better than both EM-ELM and ELM. These algorithms could also be implemented on regression problem.

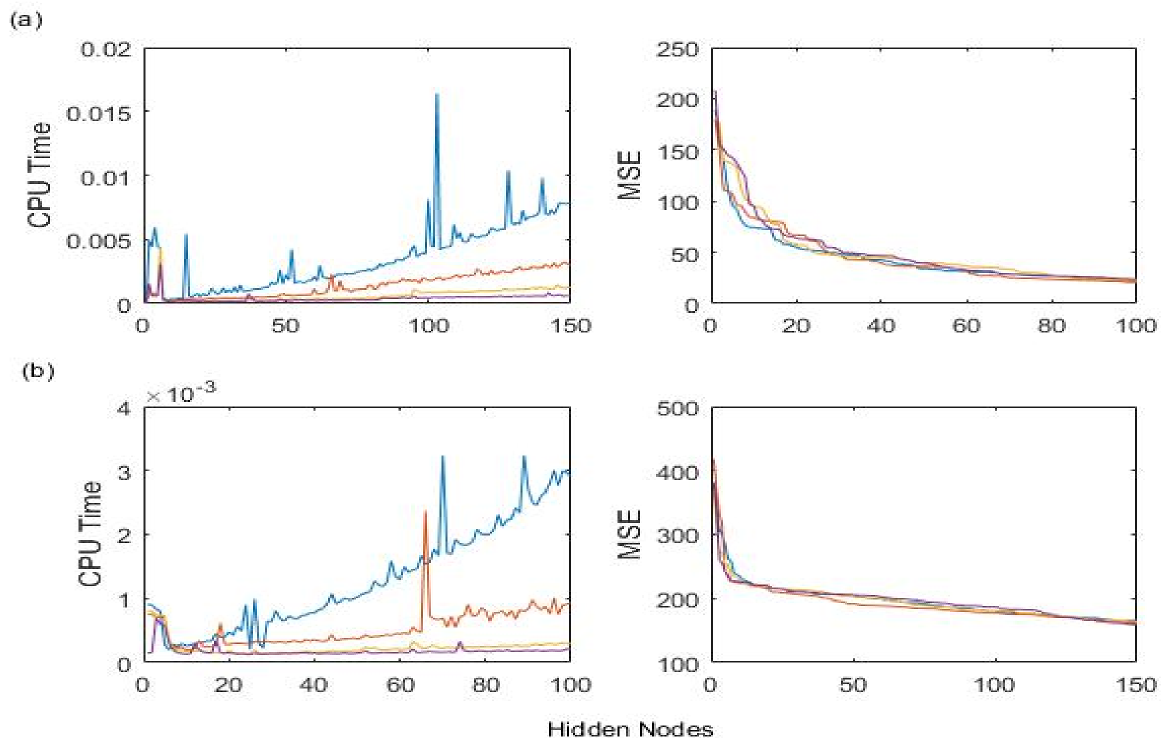

Figure 1. Results of all algorithms for CPU time and mean square error on (a) Pima and (b) Ionosphere datasets.

## 5. Conclusions

In this study, a method for incremental ELM was proposed based on pseudoinverse matrix decomposition. This method automatically determines the number of hidden nodes in ELM. During the growth of network, the output weights is incrementally updated. Findings show that new method is slightly faster than its recent version which is based on QR factorization and has similar generalization performance.

## Conflict of Interest

No potential conflict of interest relevant to this article was reported.

## References

[1] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006. http://dx.doi.org/10.1109/TNN.2006.880583

[2] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513-529, 2012. http://dx.doi.org/10.1109/TSMCB.2011.2168604

[3] ] Z. Bai, G. B. Huang, D. Wang, H. Wang, and M. B. Westover, "Sparse extreme learning machine for classification," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1858-1870, 2014. http://dx.doi.org/10.1109/TCYB.2014.2298235

[4] J. Luo, C. M. Vong, and P. K. Wong, "Sparse Bayesian extreme learning machine for multi-classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 836-843, 2014. http://dx.doi.org/10.1109/TNNLS.2013.2281839

[5] W. Zong, G. B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229-242, 2013. http://dx.doi.org/10.1016/j.neucom.2012.08.010

[6] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural

networks," in *Proceedings of IEEE International Joint Conference on Neural Networks,* Budapest, Hungary, 2004, pp. 985-990. http://dx.doi.org/10.1109/IJCNN.2004.1380068

[7] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: optimally pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158-162, 2010. http://dx.doi.org/10.1109/TNN.2009.2036259

[8] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on MapReduce," *Neurocomputing*, vol. 149, pp. 456-463, 2015. http://dx.doi.org/10.1016/j.neucom.2014.01.070

[9] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405-2417, 2014. http://dx.doi.org/10.1109/TCYB.2014.2307349

[10] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006. http://dx.doi.org/10.1109/TNN.2006.875977

[11] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006. http://dx.doi.org/10.1016/j.neucom.2005.12.126

[12] G. Feng, G. B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1352-1357, 2009. http://dx.doi.org/10.1109/TNN.2009.2024147

[13] Y. Ye and Y. Qin, "QR factorization based incremental extreme learning machine with growth of hidden nodes," *Pattern Recognition Letters*, vol. 65, pp. 177-183, 2015. http://dx.doi.org/10.1016/j.patrec.2015.07.031

[14] T. N. E. Greville, "Some applications of the pseudoinverse of a matrix," *SIAM Review*, vol. 2, no. 1, pp. 15-22, 1960. http://dx.doi.org/10.1137/1002004

**Peyman Hosseinzadeh Kassani** received his master degree in computer science from University of Economic Sciences, Tehran, Iran in 2012 and became a Member of IEEE in 2013. He is currently doing his Ph.D. at Yonsei University, Seoul, South Korea. He is with computational intelligence laboratory in Department of Electrical and Electronics Engineering of Yonsei University. He has worked on object detection and pattern classification. His general interests lie in machine learning and pattern recognition and their application to computer vision.

**Euntai Kim** was born in Seoul, Korea, in 1970. He received B.S., M.S., and Ph.D. degrees in Electronic Engineering, all from Yonsei University, Seoul, Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2002, he was a Full-Time Lecturer in the Department of Control and Instrumentation Engineering, Hankyong National University, Kyonggi-do, Korea. Since 2002, he has been with the faculty of the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was a Visiting Scholar at the University of Alberta, Edmonton, AB, Canada, in 2003, and also was a Visiting Researcher at the Berkeley Initiative in Soft Computing, University of California, Berkeley, CA, USA, in 2008. His current research interests include computational intelligence and statistical machine learning and their application to intelligent robotics, unmanned vehicles, and robot vision.