12-2008

A fast pruned-extreme learning machine for classification problem

Hai-Jun RONG
Nanyang Technological University

Yew-Soon ONG
Nanyang Technological University

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Zexuan ZHU
Nanyang Technological University

Citation

RONG, Hai-Jun; ONG, Yew-Soon; TAN, Ah-hwee; and ZHU, Zexuan. A fast pruned-extreme learning machine for classification problem. (2008). *Neurocomputing*. 72, (1-3), 359-366.
Available at: https://ink.library.smu.edu.sg/sis_research/5210

Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Neurocomputing 72 (2008) 359–366
NEUROCOMPUTING
www.elsevier.com/locate/neucom

A fast pruned-extreme learning machine for classification problem

Hai-Jun Rong, Yew-Soon Ong*, Ah-Hwee Tan, Zexuan Zhu

Division of Information Systems, School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Received 18 June 2007; received in revised form 26 December 2007; accepted 1 January 2008
Communicated by D. Wang
Available online 5 February 2008

Abstract

Extreme learning machine (ELM) represents one of the recent successful approaches in machine learning, particularly for performing pattern classification. One key strength of ELM is the significantly low computational time required for training new classifiers since the weights of the hidden and output nodes are randomly chosen and analytically determined, respectively. In this paper, we address the architectural design of the ELM classifier network, since too few/many hidden nodes employed would lead to underfitting/overfitting issues in pattern classification. In particular, we describe the proposed pruned-ELM (P-ELM) algorithm as a systematic and automated approach for designing ELM classifier network. P-ELM uses statistical methods to measure the relevance of hidden nodes. Beginning from an initial large number of hidden nodes, irrelevant nodes are then pruned by considering their relevance to the class labels. As a result, the architectural design of ELM network classifier can be automated. Empirical study of P-ELM on several commonly used classification benchmark problems and with diverse forms of hidden node functions show that the proposed approach leads to compact network classifiers that generate fast response and robust prediction accuracy on unseen data, comparing with traditional ELM and other popular machine learning approaches.

Keywords: Feedforward networks; Extreme learning machine (ELM); Pattern classification

1. Introduction

A new fast learning neural algorithm referred to as extreme learning machine (ELM) with additive hidden nodes and radial basis function (RBF) kernels has been developed for single-hidden layer feedforward networks (SLFNs) in [10,11]. ELM has been successfully applied to many real world applications [13,4,18,19,12] and has been shown to generate good generalization performance at extremely high learning speed [10–13]. Nevertheless, the number of hidden nodes to be used when designing the classifier using ELM for handling the problem in hand, remains a trial and error process.

In the context of pattern recognition, it is now widely known that a classifier network that is too small generally lacks the capability of learning the training data sufficiently well. On the other hand, a network that is too large could also overfit the training data, thus producing poor generalization performance on unseen cases. Besides, an extremely large network also brings about longer prediction responses and unnecessary requirement for large memory as well as high cost for hardware implementation. An alternative to improve generalization is to train a network that is larger than necessary and prune the unnecessary nodes during learning.

In this paper, we address the architectural design of the ELM classifier network, since too small/large hidden nodes employed would lead to underfitting/overfitting issues in pattern classification. In particular, we present a pruned-ELM (P-ELM) algorithm as a systematic and automated approach for designing ELM classifier network. P-ELM provides a systematic approach for designing the network architecture of the ELM classifier. Using statistical methods to measure the relevance of each hidden node in contributing to the prediction accuracy of the classifier, the appropriate architecture of the classifier network is then defined.

*Corresponding author.
E-mail address: ASYSOng@ntu.edu.sg (Y.-S. Ong).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.01.005

When designing a classifier, P-ELM begins with an initial large number of hidden nodes whose weight parameters are randomly assigned to obtain the hidden node response of the input vectors. Based on the relevance of each hidden node in contributing to the prediction accuracy of the classifier, irrelevant/relevant hidden nodes are pruned/kept intact. The notion of relevancy is defined by some threshold which is user-definable. Since P-ELM is sensitive to the relevance threshold level used, i.e., the choice of relevance threshold used results in classifier networks of different size and prediction accuracy, a recommended threshold base is further identified based on the Akaike information criterion (AIC).

Simulation study of the P-ELM on several commonly used classification benchmark problems for diverse forms of hidden nodes show that the proposed approach leads to compact network classifiers that generate fast response and robust prediction accuracy on unseen data when compared with the traditional ELM and other popular machine learning approaches including the backpropagation (BP) [12] and minimal resource allocating network (MRAN) [15] algorithms.

This paper is organized as follows. Section 2 provides a brief review of the ELM. The details of the proposed P-ELM algorithm is then described in Section 3. Section 4 presents a quantitative performance comparison of P-ELM to other algorithms based on commonly used classification benchmark problems. Finally, Section 5 summarizes the conclusions of the present study.

2. A brief review of ELM

This section briefly reviews the ELM proposed in [9–11]. One key principle of the ELM is that one may randomly choose and fix the hidden node parameters. After the hidden nodes parameters are chosen randomly, SLFN becomes a linear system where the output weights of the network can be analytically determined using simple generalized inverse operation of the hidden layer output matrices.

2.1. Extreme learning machine (ELM)

The output of a SLFN with $\tilde{N}$ hidden nodes (additive or RBF nodes) can be represented by

$$f_{\tilde{N}}(\mathbf{x}) = \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i G(\mathbf{x};\mathbf{c}_i,a_i), \quad \mathbf{x} \in \mathbf{R}^n, \ \mathbf{c}_i \in \mathbf{R}^n, \tag{1}$$

where $\mathbf{c}_i$ and a_i are the learning parameters of hidden nodes, $\boldsymbol{\beta}_i$ is the weight connecting the ith hidden node to the output node, and $G(\mathbf{x};\mathbf{c}_i,a_i)$ is the output of the ith hidden node with respect to the input $\mathbf{x}$. For additive hidden nodes with the sigmoid or threshold activation function $g(x): R \to R$, $G(\mathbf{x};\mathbf{c}_i,a_i)$ is given by

$$G(\mathbf{x};\mathbf{c}_i,a_i) = g(\mathbf{c}_i \cdot \mathbf{x} + a_i), \quad a_i \in R, \tag{2}$$

where $\mathbf{c}_i$ is the weight vector connecting the input layer to the ith hidden node, a_i is the bias of the ith hidden node, and $\mathbf{c}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{c}_i$ and $\mathbf{x}$ in $\mathbf{R}^n$.

For RBF hidden nodes with the Gaussian or triangular activation function $g(x): R \to R$, $G(\mathbf{x};\mathbf{c}_i,a_i)$ is given by

$$G(\mathbf{x};\mathbf{c}_i,a_i) = g(a_i\|\mathbf{x}-\mathbf{c}_i\|), \quad a_i \in R^+, \tag{3}$$

where $\mathbf{c}_i$ and a_i are the center and impact factor of ith RBF node. R^+ indicates the set of all positive real values.

For N arbitrary distinct samples $(\mathbf{x}_k,\mathbf{t}_k) \in \mathbf{R}^n \times \mathbf{R}^m$, if a SLFN with $\tilde{N}$ hidden nodes can approximate these N samples with zero error, it then implies that there exist $\boldsymbol{\beta}_i$, $\mathbf{c}_i$ and a_i such that

$$\sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i G(\mathbf{x}_k;\mathbf{c}_i,a_i) = \mathbf{t}_k, \quad k=1,\ldots,N. \tag{4}$$

Eq. (4) can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{5}$$

where

$$\mathbf{H}(\mathbf{c}_1,\ldots,\mathbf{c}_{\tilde{N}},a_1,\ldots,a_{\tilde{N}},\mathbf{x}_1,\ldots,\mathbf{x}_N) = \begin{bmatrix} G(\mathbf{x}_1;\mathbf{c}_1,a_1) & \cdots & G(\mathbf{x}_1;\mathbf{c}_{\tilde{N}},a_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_N;\mathbf{c}_1,a_1) & \cdots & G(\mathbf{x}_N;\mathbf{c}_{\tilde{N}},a_{\tilde{N}}) \end{bmatrix}_{N\times\tilde{N}}, \tag{6}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}^{\mathrm{T}} \end{bmatrix}_{\tilde{N}\times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{t}_N^{\mathrm{T}} \end{bmatrix}_{N\times m}. \tag{7}$$

$\mathbf{H}$ is called the hidden layer output matrix of the network [6,5]; the ith column of $\mathbf{H}$ is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N$ and the kth row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_k$.

Huang et al. [7,8,11] proved that for SLFNs with additive or RBF hidden nodes, one may randomly choose and fix the hidden node parameters and then analytically determine the output weights when approximating any continuous target function. Note that the proof of universal approximation is shown to be valid for SLFNs with general type of hidden nodes, i.e., in the form of sigmoid, RBF or hybrid of both [8,7].

In this regard, for N arbitrary distinct samples $(\mathbf{x}_k,\mathbf{t}_k)$, in order to obtain arbitrarily small non-zero training error, one may randomly generate $\tilde{N}$ $(\leqslant N)$ hidden nodes (with random parameters $(\mathbf{c}_i,a_i)$). Eq. (5) then becomes a linear system and the output weights $\boldsymbol{\beta}$ are estimated as

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{8}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse [16] of the hidden layer output matrix $\mathbf{H}$. Calculation of the output weights is done in a single step here. Thus this avoids any lengthy training procedure where the network parameters are adjusted iteratively with appropriately chosen control parameters (learning rate and learning epochs, etc.). The *three-step* ELM algorithm [10,11] can be summarized as follows:

ELM Algorithm. Given a training set $\aleph = \{(\mathbf{x}_k, \mathbf{t}_k)|\mathbf{x}_k \in \mathbf{R}^n, \mathbf{t}_k \in \mathbf{R}^m, k = 1, \ldots, N\}$, activation function g, and hidden nodes $\tilde{N}$,

Step 1: Randomly assign hidden node parameters $(\mathbf{c}_i, a_i)$, $i = 1, \ldots, \tilde{N}$.

Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.

Step 3: Calculate the output weight $\boldsymbol{\beta}$: $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$.

In the standard ELM algorithm proposed in [10,11], the number of the hidden nodes, $\tilde{N}$, to use is a decision to be made by the designer of the classifier network. Often a trial and error process is conducted in advance, in order to arrive at networks that produce good generalization performances. To avoid this potential erroneous and computational intensive trial and error procedure, a P-ELM algorithm is proposed as a systematic and automated approach for defining the hidden node size $\tilde{N}$ in the design of classifier networks. In what follows, we present a detailed description of the proposed P-ELM.

3. Pruned-ELM

The main idea of the proposed P-ELM lies in identifying the degree of relevance between the hidden nodes and the class labels with the use of some statistical measure and then removing the irrelevant or low relevant hidden nodes so as to arrive at compact networks while keeping the generalization ability of the network uncompromised.

In the first step of the proposed P-ELM, an initial network of $(\tilde{N})$ hidden node size is assigned. Based on the standard methodology of ELM, all parameters of the hidden nodes, i.e., $((\mathbf{c}_i, a_i), i = 1, \ldots, \tilde{N})$ are then randomly assigned. Based on the assigned parameters, the responses of the hidden nodes (represented here as matrix $\mathbf{H}$) are subsequently obtained. Together with the labelled training data and the responses of the hidden nodes ($\mathbf{H}$), the statistical relevance of each hidden node in contributing to the true class label is then identified. Here we investigate two statistical criteria, namely, the Chi-squared (χ^2) [14,20] and information gain (IG) [1,17] measures, for revealing the statistical relevance of the hidden nodes in arriving at the true class labels. Note that the responses of the hidden nodes are discretized based on the entropy-based discretization method [3] before the statistical methods are put in place.

3.1. Chi-squared (χ^2)

The χ^2 method can be used to measure the association between the class labels and individual hidden node $h_i (i = 1, 2, \ldots, \tilde{N})$. It is defined as

$$\chi^2(h_i) = \sum_{l=1}^{d}\sum_{j=1}^{r}\frac{(A_{lj} - E_{lj})^2}{E_{lj}},$$

$$E_{lj} = \frac{R_l * Z_j}{N}, \quad R_l = \sum_{j=1}^{r} A_{lj},$$

$$Z_j = \sum_{l=1}^{d} A_{lj}, \quad N = \sum_{l=1}^{d}\sum_{j=1}^{r} A_{lj}, \tag{9}$$

where d is the number of discrete bins for individual hidden node h_i and r is the number of class labels; A_{lj} represents the cardinality of bin l and class label j and E_{lj} is the expected cardinality of A_{lj}. Based on the χ^2 measure, the hidden nodes are sorted in descending order. Note that the larger the χ^2 value, the higher relevance is the hidden node to the class.

3.2. Information gain (IG)

The IG criterion represents an alternative for measuring the reduction in entropy of the class $\mathbf{t}(t_1, t_2, \ldots, t_r)$, which reflects the additional information provided by each hidden node $h_i (i = 1, 2, \ldots, \tilde{N})$. It is defined as,

$$IG(h_i) = -\sum_{j=1}^{r} p(t_j)\log_2 p(t_j) + \sum_{l=1}^{d} p(h_{i,l})\sum_{j=1}^{r} p(t_j|h_{i,l})\log_2 p(t_j|h_{i,l}), \tag{10}$$

where r is the number of class labels and d is the number of discrete bins for hidden node h_i; $p(t_j)$ is the prior probability for class j and $p(h_{i,l})$ is the prior probability for discrete bin l in hidden node h_i; $p(t_j|h_{i,l})$ is the probability for t_j conditioned on $h_{i,l}$. Subsequently, the hidden nodes are sorted in descending order, based on the IG derived. A hidden node of higher IG is preferred since it indicates a greater importance in contributing to the class label.

Using the statistical relevance measures obtained, irrelevant/relevant hidden nodes are removed/kept intact. Here a hidden node is considered as irrelevant in contributing to the true class label if the relevance measures fall below the normalized relevance threshold parameter, $\gamma \in [0, 1.0]$, which is user-defined. Note that the choice of the relevance threshold directly impacts the structure and generalization ability of the resultant P-ELM network. As opposed to a trial and error process in defining the relevance threshold, the final network structure complexity and generalization ability are systematically determined from a base set of potential relevance threshold values, i.e., γ_i, $i = 1, 2, \ldots, q$, using the AIC. Generally, γ_i is sampled uniformly and q defines the size of the base set required. Note that in the paper the optimal relevance threshold for each problem considered is determined using the validation data drawn from the training data. The AIC is given by,

$$\mathrm{AIC}(i) = 2N_{\mathrm{val}}\ln(\delta_i^2/N_{\mathrm{val}}) + S_i, \quad i = 1, 2, \ldots, q, \tag{11}$$

where N_{val} is the number of validation data; δ_i is the classification error for the ith relevance threshold and equals to $(1-o_i)*N_{\text{val}}$; o_i is correct validation classification rate and S_i is the hidden node size for the ith relevance threshold. AIC aims to achieve a trade-off between prediction accuracy and the structural complexity of the final network.

Here, we provide an outline of the P-ELM algorithm:

P-ELM Algorithm. Given a training set $\aleph = \{(\mathbf{x}_k, \mathbf{t}_k) | \mathbf{x}_k \in \mathbf{R}^n, \mathbf{t}_k \in \mathbf{R}^m, k = 1, \ldots, N\}$, activation function g, an initial larger hidden node size than necessary, $\tilde{N}$, and a relevance threshold base $\boldsymbol{\gamma}(\gamma_1, \gamma_2, \ldots, \gamma_q)$,

(1) Separate the training set into two non-overlapping subsets for learning and validation.
(2) Randomly assign hidden node parameters $(\mathbf{c}_i, a_i)$, $i = 1, \ldots, \tilde{N}$.
(3) Calculate the hidden layer output matrix $\mathbf{H}$ using the learning subset.
(4) Calculate the statistical relevance for the hidden nodes using the χ^2 or IG and sort them in descending order.
(5) For each relevance threshold $\gamma_i (i = 1, 2, \ldots, q)$.
   - Identify the subset of crucial hidden nodes S_i that satisfy the relevance threshold γ_i and determine the validation accuracy o_i using the validation subset.
   - Calculate the AIC$(i) = f(S_i, o_i)$ using Eq. (11).
(6) Select S^* according to min(AIC).
(7) Retrain network S^* with the whole training set.
   - Calculate the hidden layer output matrix $\mathbf{H}^*$ using the training set.
   - Calculate the output weight $\boldsymbol{\beta}^*$: $\boldsymbol{\beta}^* = (\mathbf{H}^*)^{\dagger}\mathbf{T}$.
(8) Evaluate the performance of S^* on unseen testing data set.

4. Performance evaluation

In this section, the numerical performance of the proposed P-ELM is evaluated in details with comparison to several state-of-the-art machine learning approaches including the standard ELM, BP [12] and MRAN [15] for pattern classification. Eight real-world classification

Table 1
Specification of real-world classification benchmark problems

Data sets	No. of attributes	No. of classes	No. of observations	
			Training	Testing
Page blocks	10	5	2700	2773
Image segmentation	19	7	1100	1210
Satellite image	36	6	4400	2035
Spam emails	57	2	3000	1601
Glass	9	7	110	104
Liver disorders	6	2	170	175
Vehicle	18	4	420	426
Letter recognition	16	26	10,000	10,000

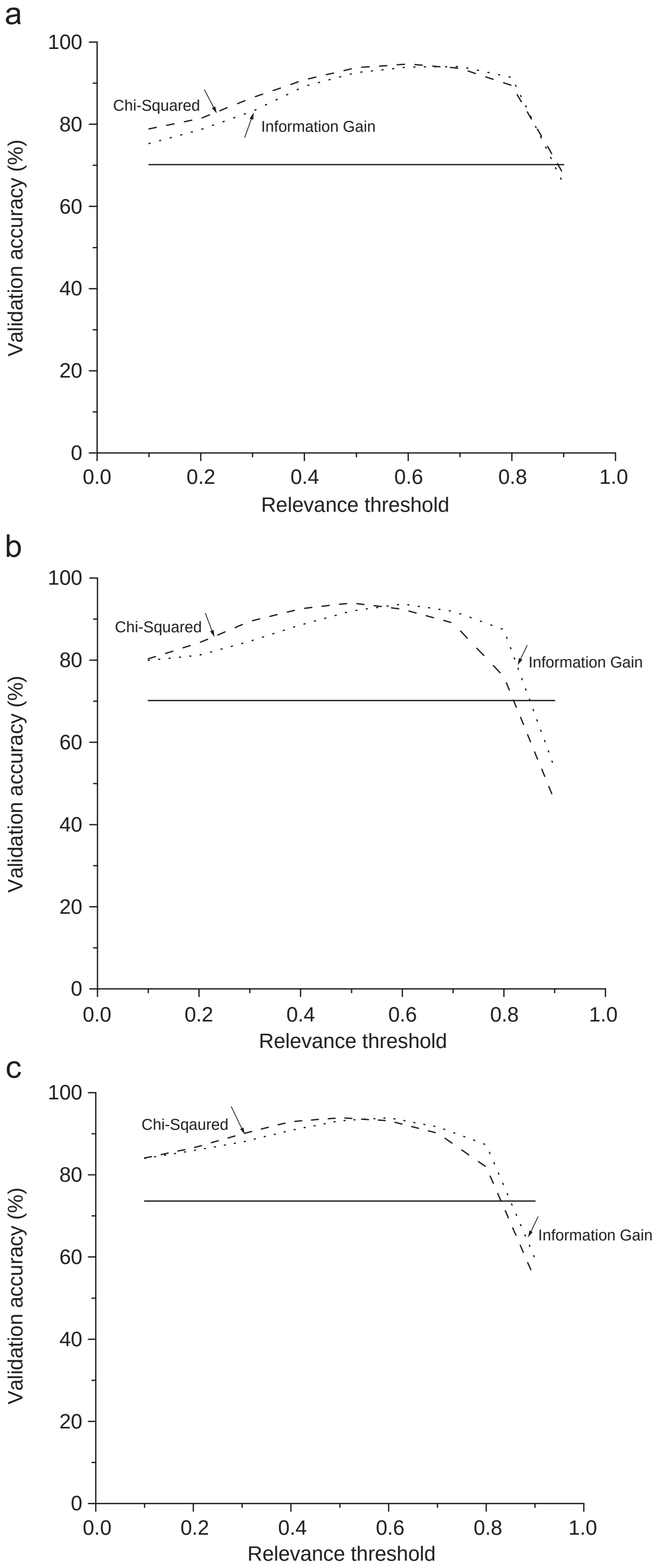

Fig. 1. Effect of the relevance threshold on the validation accuracy using χ^2 and IG methods on the image segmentation problem: (a) sigmoid hidden nodes, (b) RBF hidden nodes, (c) hybrid sigmoid-RBF hidden nodes.

problems from UCI ML repository [2] are considered for verifying the performance of P-ELM, which are summarized in Table 1 with separate training and testing data sets. Subsequently, the training data are further split randomly as 75% and 25% learning and validation data sets, respectively. The χ^2 and IG statistical methods used for pruning the insignificant hidden nodes are denoted as P-ELM1 and P-ELM2, respectively. Here the number of hidden nodes used in the standard ELM is chosen in a trial and error manner since this is the common practice by the general community when using ELM. In our experimental study, we consider the number of hidden nodes in the range of $[10, 500]$ and a step size of 10 nodes. This defines the overall training time of the standard ELM.

The experimental results reported here are based on the average of 50 independent trials. All simulations have been conducted within the MATLAB 7.1 environment on a Pentium IV 1.7 GHZ CPU and 512 M RAM personal computer.

4.1. Study on relevance threshold

Here we consider the image segmentation data set for studying the effects of the relevance threshold used on the validation accuracy of P-ELM. Fig. 1 illustrates the impact of the relevance threshold on the validation accuracy based on the χ^2 or IG criterion for three kinds of hidden nodes, i.e., sigmoid, RBF and their combination. It should be noted that the relevance between hidden nodes and the class labels is normalized to $[0, 1]$, which also maintains the relevance threshold at $[0, 1]$. A threshold value of 0 implies that no hidden nodes are removed while the other extreme indicates all hidden nodes are removed. Hence it makes sense to consider having a relevance threshold range of $[0.1, 0.9]$ in the proposed P-ELM. Further, a relevance threshold step size of 0.1 is considered for investigating its impacts on validation accuracy. From Fig. 1, it is observed that an initial large hidden node size would result in poor validation accuracy on unseen data, i.e., this is represented by the solid line, due to the effects of overfitting on the training data. On the other hand, with the use of χ^2 or IG criterion for pruning the irrelevant nodes in P-ELM, the validation accuracy on the unseen data, i.e., represented by the dashed and dotted lines, respectively, is shown to have improved significantly. Nevertheless, the results further illustrate that with increasing relevance threshold values, the validation accuracy also improves but underfitting may occur at some turning point. From the results in Fig. 1 and other data sets where similar observations are made, it becomes possible to generalize that the range of $[0.1, 0.9]$ is effective for generating compact networks that generates robust prediction accuracy. Hence, as a rule of thumb, we suggest using a relevance threshold value, γ, in the range of $[0.1, 0.9]$ at a step size of 0.1.

Table 2
Effect of initial hidden node size on the final ELM network structure and testing accuracy, considering the image segmentation problem

Node types	Methods	No. of initial nodes	No. of final nodes	Testing (%)		Training time (s)
				Rate	Dev.	
Sigmoid	P-ELM1	500	198	94.830	0.622	16.220
		1000	181	94.795	0.616	72.787
		1100	185	94.704	0.534	92.757
		2000	186	94.864	0.663	194.14
	P-ELM2	500	189	94.654	0.705	18.263
		1000	199	94.895	0.591	84.733
		1100	207	94.821	0.639	104.99
		2000	205	94.932	0.619	223.76
RBF	P-ELM1	500	203	94.352	0.761	18.911
		1000	201	94.988	0.707	79.060
		1100	198	94.803	0.745	82.988
		2000	197	94.557	0.695	210.33
	P-ELM2	500	225	94.364	0.808	22.009
		1000	202	94.630	0.791	89.252
		1100	171	94.377	0.797	94.220
		2000	191	94.297	0.762	235.66
Hybrid	P-ELM1	500	213	94.747	0.903	18.214
		1000	223	94.963	0.713	74.423
		1100	201	94.803	0.834	91.845
		2000	216	94.735	0.731	204.91
	P-ELM2	500	197	94.185	0.736	20.092
		1000	208	94.932	0.822	87.145
		1100	185	94.525	0.744	102.96
		2000	201	94.462	0.701	228.42

4.2. Study on effect of hidden node initialization

To further study the robustness of the proposed P-ELM methodology for generating compact networks, we also investigate the possible impact of the initial number of hidden nodes used on the resultant network structure complexity and prediction accuracy. Table 2 tabulates the resultant network complexity and the testing accuracy for different forms of hidden nodes. Note that as a rule of thumb the number of hidden nodes is initialized as equal to the size of the training data as considered in the present study. The results indicate that the generalization performance and network complexity of the ELM network remain relatively stable regardless of the initial hidden nodes used, i.e., for the sake of convenience one should choose to initialize the number of initial hidden nodes as equal to the size of the training data. This also implies that the proposed P-ELM algorithm is insensitive to the initial number of hidden nodes used. Nevertheless we would like to highlight that a large initial hidden nodes generally lead to increased computational burden. In our study, 500 initial hidden nodes appears to be sufficient for the problems considered.

4.3. Performance comparison with other machine learning approaches

Here, we present the performance of the proposed P-ELM in terms of prediction accuracy on unseen data, training wall-clock time and hidden node size of final ELM network found for diverse forms of hidden nodes, i.e., sigmoid, RBF and their combination. The core purpose is to study the performance of the P-ELM against some existing state-of-the-art machine learning approaches on real-world classification problems. For the sake of brevity, the algorithms considered in the present study for comparison are listed in Table 3.

The classification performance of the machine learning algorithms together with those obtained by the P-ELMs on the real world classification benchmark problems are then tabulated in Tables 4–6. In comparison to BP, P-ELM$_{\mathrm{Sig}}$ arrives at competitive prediction accuracy but doing so at significantly lower training time. Note that in what is reported in Table 4, the training time of BP excludes the computational time used to choose the appropriate hidden nodes for the problems considered. Similarly, based on the results in Table 5, P-ELM$_{\mathrm{Rbf}}$ also displays better computational efficiency and generalization ability than the MRAN. Note that the result for MRAN on the letter recognition problem is excluded since this method becomes computationally intractable when dealing with large data set. In addition, the results in the tables reported illustrate the low training time cost of P-ELMs. In contrast to the standard ELM, which involves a trial-and-error process to identify a suitable architecture of the network, P-ELMs search for a suitable network architecture, i.e., identifying an appropriate number of hidden nodes for the data set in hand, based on statistical information, hence generating the significant saving in training time. Furthermore, P-ELMs produce significantly more compact networks compared with the standard ELM through the removal of irrelevant hidden nodes. In comparison to the standard ELM, P-ELMs is also unaffected by the functional form of the hidden node used. Hence P-ELMs demonstrate

Table 3
Specification of the machine learning algorithms considered and their respective hidden nodes

Types of hidden nodes	Algorithms		
Sigmoid	P-ELM$_{\mathrm{Sig}}$	ELM$_{\mathrm{Sig}}$	BP
RBF	P-ELM$_{\mathrm{Rbf}}$	ELM$_{\mathrm{Rbf}}$	MRAN
Hybrid	P-ELM$_{\mathrm{Hyb}}$	ELM$_{\mathrm{Hyb}}$	–

Table 4
The performance comparison between P-ELM$_{\mathrm{Sig}}$, ELM$_{\mathrm{Sig}}$ and BP algorithms

Data sets	Methods	No. of nodes	Testing (%)		Training time (s)
			Rate	Dev.	
Page blocks	P-ELM1$_{\mathrm{Sig}}$	**92**	95.745	0.296	**24.987**
	P-ELM2$_{\mathrm{Sig}}$	**100**	95.800	0.327	**26.863**
	ELM$_{\mathrm{Sig}}$	160	95.722	0.333	92.285
	BP	10	95.515	0.665	63.983
Image segmentation	P-ELM1$_{\mathrm{Sig}}$	**198**	94.830	0.622	**16.220**
	P-ELM2$_{\mathrm{Sig}}$	**189**	94.654	0.705	**18.263**
	ELM$_{\mathrm{Sig}}$	210	94.402	0.782	65.220
	BP	100	94.030	0.886	1030.7
Satellite image	P-ELM1$_{\mathrm{Sig}}$	**397**	89.955	0.563	**49.533**
	P-ELM2$_{\mathrm{Sig}}$	**409**	89.938	0.523	**52.152**
	ELM$_{\mathrm{Sig}}$	470	89.735	0.605	150.76
	BP	100	89.440	0.824	228.48
Spam emails	P-ELM1$_{\mathrm{Sig}}$	**184**	91.090	0.415	**12.628**
	P-ELM2$_{\mathrm{Sig}}$	**213**	91.593	0.571	**12.589**
	ELM$_{\mathrm{Sig}}$	380	91.037	0.598	103.10
	BP	50	91.836	0.745	4641.9
Glass	P-ELM1$_{\mathrm{Sig}}$	**18**	65.136	3.315	**0.5681**
	P-ELM2$_{\mathrm{Sig}}$	**20**	65.079	3.937	**0.6461**
	ELM$_{\mathrm{Sig}}$	30	64.923	3.656	3.6555
	BP	10	65.154	4.728	0.8938
Liver disorders	P-ELM1$_{\mathrm{Sig}}$	**14**	68.771	3.640	**0.3336**
	P-ELM2$_{\mathrm{Sig}}$	**17**	68.957	3.484	**0.3141**
	ELM$_{\mathrm{Sig}}$	20	68.891	3.271	6.3195
	BP	20	68.066	3.870	0.6282
Vehicle	P-ELM1$_{\mathrm{Sig}}$	**67**	79.254	2.246	**2.6883**
	P-ELM2$_{\mathrm{Sig}}$	**74**	79.138	2.351	**2.7109**
	ELM$_{\mathrm{Sig}}$	110	79.176	2.402	31.973
	BP	25	79.502	3.685	66.089
Letter recognition	P-ELM1$_{\mathrm{Sig}}$	**483**	89.669	0.212	**214.03**
	P-ELM2$_{\mathrm{Sig}}$	**480**	89.652	0.199	**214.63**
	ELM$_{\mathrm{Sig}}$	500	89.057	0.221	317.95
	BP	100	85.390	1.636	3977.2

Table 5
The performance comparison between P-ELM$_{\text{Rbf}}$, ELM$_{\text{Rbf}}$ and MRAN algorithms

Data sets	Methods	No. of nodes	Testing (%)		Training time (s)
			Rate	Dev.	
Page blocks	P-ELM1$_{\text{Rbf}}$	**155**	95.915	0.353	**28.410**
	P-ELM2$_{\text{Rbf}}$	**144**	95.871	0.364	**29.064**
	ELM$_{\text{Rbf}}$	160	95.806	0.364	237.41
	MRAN	33	93.870	1.023	1100.0
Image segmentation	P-ELM1$_{\text{Rbf}}$	**203**	94.352	0.761	**18.911**
	P-ELM2$_{\text{Rbf}}$	**225**	94.364	0.808	**22.009**
	ELM$_{\text{Rbf}}$	310	94.280	0.911	166.334
	MRAN	48	92.284	0.837	4893.5
Satellite image	P-ELM1$_{\text{Rbf}}$	**332**	89.385	0.586	**57.810**
	P-ELM2$_{\text{Rbf}}$	**355**	89.533	0.617	**60.471**
	ELM$_{\text{Rbf}}$	470	89.440	0.620	588.55
	MRAN	10	86.138	1.759	727.55
Spam emails	P-ELM1$_{\text{Rbf}}$	**213**	90.818	0.745	**19.759**
	P-ELM2$_{\text{Rbf}}$	**242**	90.793	0.646	**19.402**
	ELM$_{\text{Rbf}}$	410	90.605	0.769	520.59
	MRAN	16	90.319	1.759	1053.5
Glass	P-ELM1$_{\text{Rbf}}$	**20**	64.604	4.125	**0.6820**
	P-ELM2$_{\text{Rbf}}$	**21**	64.844	3.683	**0.7703**
	ELM$_{\text{Rbf}}$	30	64.923	4.393	6.5820
	MRAN	19	63.231	4.257	4.7281
Liver disorders	P-ELM1$_{\text{Rbf}}$	**19**	68.714	3.814	**0.4992**
	P-ELM2$_{\text{Rbf}}$	**16**	68.143	4.481	**0.5258**
	ELM$_{\text{Rbf}}$	30	68.429	3.954	10.031
	MRAN	19	60.286	4.061	1.2922
Vehicle	P-ELM1$_{\text{Rbf}}$	**84**	79.967	1.348	**3.2984**
	P-ELM2$_{\text{Rbf}}$	**73**	79.504	1.889	**3.2000**
	ELM$_{\text{Rbf}}$	120	79.980	1.863	53.342
	MRAN	32	71.354	1.915	245.77
Letter recognition	P-ELM1$_{\text{Rbf}}$	**446**	89.098	0.257	**228.89**
	P-ELM2$_{\text{Rbf}}$	**447**	89.027	0.257	**232.81**
	ELM$_{\text{Rbf}}$	500	88.120	0.346	988.20
	MRAN	N.A.	N.A.	N.A.	N.A.

Table 6
The performance comparison between P-ELM$_{\text{Hyb}}$ and ELM$_{\text{Hyb}}$ algorithms

Data sets	Methods	No. of nodes	Testing (%)		Training time (s)
			Rate	Dev.	
Page blocks	P-ELM1$_{\text{Hyb}}$	**132**	95.591	0.404	**26.141**
	P-ELM2$_{\text{Hyb}}$	**124**	95.602	0.397	**27.321**
	ELM$_{\text{Hyb}}$	170	95.531	0.419	223.18
Image segmentation	P-ELM1$_{\text{Hyb}}$	**213**	94.747	0.903	**18.214**
	P-ELM2$_{\text{Hyb}}$	**197**	94.185	0.736	**20.092**
	ELM$_{\text{Hyb}}$	230	94.461	0.984	150.21
Satellite image	P-ELM1$_{\text{Hyb}}$	**357**	89.693	0.493	**56.112**
	P-ELM2$_{\text{Hyb}}$	**371**	89.858	0.582	**56.013**
	ELM$_{\text{Hyb}}$	470	89.733	0.567	474.72
Spam emails	P-ELM1$_{\text{Hyb}}$	**221**	90.937	0.555	**17.781**
	P-ELM2$_{\text{Hyb}}$	**201**	91.124	0.487	**17.370**
	ELM$_{\text{Hyb}}$	390	90.615	0.571	389.14
Glass	P-ELM1$_{\text{Hyb}}$	**20**	65.004	4.064	**0.6766**
	P-ELM2$_{\text{Hyb}}$	**20**	65.100	4.002	**0.7617**
	ELM$_{\text{Hyb}}$	30	65.067	4.725	6.0234
Liver disorders	P-ELM1$_{\text{Hyb}}$	**13**	68.086	3.687	**0.4945**
	P-ELM2$_{\text{Hyb}}$	**12**	68.114	3.620	**0.4805**
	ELM$_{\text{Hyb}}$	20	68.471	3.847	9.6570
Vehicle	P-ELM1$_{\text{Hyb}}$	**73**	79.280	2.093	**2.8828**
	P-ELM2$_{\text{Hyb}}$	**75**	79.723	2.034	**3.0844**
	ELM$_{\text{Hyb}}$	130	79.608	2.111	50.134
Letter recognition	P-ELM1$_{\text{Hyb}}$	**447**	89.181	0.318	**225.61**
	P-ELM2$_{\text{Hyb}}$	**458**	89.225	0.387	**231.41**
	ELM$_{\text{Hyb}}$	500	88.840	0.351	889.02

excellent robustness in the generalization ability of the final network. As far as the χ^2 and IG statistical methods are concerned, there is little significant differences in the testing accuracy and network structure attained by either.

5. Conclusions

In this paper, a P-ELM algorithm is proposed to systematically determine the number of the hidden nodes during learning by using statistical methods to remove the irrelevant or lowly relevant hidden nodes. A performance comparison of P-ELM with other well-known learning algorithms including the standard ELM, BP and MRAN has been carried out on benchmark problems of some real-world classification problems. To be precise, in comparison to the BP and MRAN algorithms, P-ELM achieves competitive or improved testing accuracy at significantly lower training time incurred on the classification problems considered. Further, in contrast to the standard ELM algorithm, P-ELM does not select the hidden nodes in a trial and error manner, thus producing compact network structure that generates competitive prediction accuracy on unseen cases.

Acknowledgments

This work is funded in part under the A*STAR SERC Grant no. 052 015 0024 administered through the National Grid office. The authors wish to thank the editors and anonymous referees for their constructive comments on an earlier draft of this paper.

References

[1] R.B. Ash, Information Theory, Wiley, 1965.

[2] C. Blake, C. Merz, UCI repository of machine learning databases [online], in ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩, Department of

Information and Computer Sciences, University of California, Irvine, USA, 1998.

[3] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, San Francisco, CA, Morgan Kaufmann, Los Altos, CA, 1993, pp. 1022–1029.

[4] S.D. Handoko, K.C. Keong, Y.-S. Ong, G.L. Zhang, V. Brusic, Extreme learning machine for predicting HLA-peptide binding, Lecture Notes in Computer Science, vol. 3973, 2006, pp. 716–721.

[5] G.-B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Networks 14 (2) (2003) 274–281.

[6] G.-B. Huang, H.A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Trans. Neural Networks 9 (1) (1998) 224–229.

[7] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[8] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.

[9] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine, in: The IASTED International Conference on Computational Intelligence (CI 2005), Calgary, Canada, July 4–6, 2005.

[10] G.-B. Huang, Q.-Y. Zhu, K.Z. Mao, C.-K. Siew, P. Saratchandran, N. Sundararajan, Can threshold networks be trained directly?, IEEE Trans. Circuits Systems II 53 (3) (2006) 187–191.

[11] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.

[12] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Real-time learning capability of neural networks, IEEE Trans. Neural Networks 17 (4) (2006) 863–878.

[13] N.-Y. Liang, P. Saratchandran, G.-B. Huang, N. Sundararajan, Classification of mental tasks from EEG signals using extreme learning machine, Int. J. Neural Systems 16 (1) (2006) 29–38.

[14] H.-Q. Liu, J.-Y. Li, L.-S. Wong, A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns, Genome Informatics 13 (2002) 51–60.

[15] Y.-W. Lu, N. Sundararajan, P. Saratchandran, A sequential learning scheme for function approximation and using minimal radial basis neural networks, Neural Comput. 9 (2) (1997) 461–478.

[16] C. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Applications, Wiley, New York, 1971.

[17] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, second ed., Morgan Kaufmann, San Francisco, 2005.

[18] J.-X. Xu, W. Wang, J.C.H. Goh, G. Lee, Internal model approach for gait modeling and classification, in: The 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), Shanghai, China, September 1–4, 2005.

[19] C.-W.T. Yeu, M.-H. Lim, G.-B. Huang, A. Agarwal, Y.-S. Ong, A new machine learning paradigm for terrain reconstruction, IEEE Geosci. Remote Sensing Lett. 3 (3) (2006) 382–386.

[20] Z.-X. Zhu, Y.-S. Ong, M. Dash, Wrapper-filter feature selection algorithm using a memetic framework, IEEE Trans. Systems, Man Cybern., Part B 37 (1) (2007) 70–76.

Hai-Jun Rong received the B.Eng. from Xi'an Technological University, P.R. China, in 2000 and the M.Eng. from Xi'an JiaoTong University, P.R. China, in 2003. Currently she is studying for Ph.D. degree and working as a Research Associate in Nanyang Technological University, Singapore. Her research interests are in neural networks, fuzzy systems and pattern recognition.

Yew-Soon Ong received his B.Eng. and M.Eng. degrees in Electrical and Electronics Engineering from Nanyang Technology University, Singapore. He then joined the Computational Engineering and Design Center, University of Southampton, UK, where he completed his Ph.D. degree. Currently, he is Assistant Professor and Deputy Director of Emerging Research Lab of the School of Computer Engineering at Nanyang Technological University, Singapore. His research interests lie in computational intelligence, spanning: memetic algorithms, evolutionary design, neural networks and grid computing. He is currently an Associate Editor of the IEEE Transactions on Systems, Man and Cybernetics (Part B), Soft Computing Journal and International Journal of Systems Science.

Ah-Hwee Tan is Associate Professor and Director of Emerging Research Lab of the School of Computer Engineering at Nanyang Technological University. He is also a Faculty Associate of A*STAR Institute for Infocomm Research, where he was formally the manager of the Text Mining and Intelligent Cyber Agents groups and successfully commercialized a suite of document analysis and text mining technologies. Dr. Tan received his Ph.D. in Cognitive and Neural Systems from Boston University and B.Sc. (1st Class Hons) and M.Sc. in Computer Science from the National University of Singapore. His current research areas include cognitive and neural systems, intelligent agents, learning, and information mining. He holds several patents and has published over sixty technical papers in books, journals, and conferences. He regularly chairs international events and serves on the program committees of numerous conferences. He is on the editorial board of *Applied Intelligence* and co-edited a special issue on *Text and Web Mining*. He is a senior member of IEEE and a recipient of many awards, including Tan Kah Kee Young Inventors' Award (Silver), KRDL High Achiever Award, Optimal Gold Award, and Tan Chin Tuan Exchange Fellowship.

Zexuan Zhu received the B.S. degree from the Department of Computer Science and Engineering of Fudan University, China, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests lie in bioinformatics, evolutionary computation and machine learning.