International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems

# An Improved Algorithm for Pruning Extreme Learning Machine

--Manuscript Draft--

| | |
|---|---|
| **Manuscript Number:** | IJUFKS-D-14-00134 |
| **Full Title:** | An Improved Algorithm for Pruning Extreme Learning Machine |
| **Article Type:** | Research Paper |
| **Keywords:** | Extreme learning machine; Pruning; Architecture selection; Tolerance rough sets. |
| **Abstract:** | Extreme learning machine (ELM) is an efficient training algorithm for single-hidden layer feed-forward neural networks (SLFNs). Two pruned-ELM named P-ELM1 and P-ELM2 are proposed by Rong et al. P-ELM1 and P-ELM2 employ $\chi^{2}$ and information gain to measure the association between the class labels and individual hidden node respectively. The entropy-based discretization preprocess is introduced into the pruned-ELM for estimating the $\chi^{2}$ and information gain. The discretization will inevitably lead to information loss, and result in high computational complexity. In order to deal with this problem, based on tolerance rough sets, this paper proposed an improved pruned-ELM algorithm, which can overcome the drawbacks mentioned above. Experimental results on 8 UCI data sets show that the improved algorithm outperforms the pruned-ELM in computational complexity and testing accuracy. |

International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems


# An Improved Algorithm for Pruning Extreme Learning Machine

Junhai Zhai, Qingyan Shao, Xizhao Wang
*Key Lab. of Machine Learning and Computational Intelligence, College of Mathematics and Computer Science, Hebei University, Baoding, 071002, Hebei, China*

Received (received date)
Revised (revised date)

Extreme learning machine (ELM) is an efficient training algorithm for single-hidden layer feed-forward neural networks (SLFNs). Two pruned-ELM named P-ELM1 and P-ELM2 are proposed by Rong et al. P-ELM1 and P-ELM2 employ $\chi^2$ and information gain to measure the association between the class labels and individual hidden node respectively. The entropy-based discretization preprocess is introduced into the pruned-ELM for estimating the $\chi^2$ and information gain. The discretization will inevitably lead to information loss, and result in high computational complexity. In order to deal with this problem, based on tolerance rough sets, this paper proposed an improved pruned-ELM algorithm, which can overcome the drawbacks mentioned above. Experimental results on 8 UCI data sets show that the improved algorithm outperforms the pruned-ELM in computational complexity and testing accuracy.

*Keywords*: Extreme learning machine; Pruning; Architecture selection; Tolerance rough sets.

## 1. Introduction

As a simple and efficient training algorithm for single-hidden layer feed-forward neural networks (SLFNs), Extreme learning machine (ELM)[1,2] was proposed by Huang et al. ELM first randomly generates the weights of input layer and biases of hidden nodes and then analytically determines the weights of output layer. Huang et al.[3] have proved that ELM has universal approximation capability and good generalization performance. The key superiority of ELM is that it needs no iterations, which dramatically reduces the computational time for training the model comparing with other classical methods (such as back-propagation algorithm (BP)[4] and support vector machine (SVM)[5]). However, ELM does not provide an effective solution for architecture selection of SLFNs and it is usually pre-determined by a trial and error method, which may be tedious in some applications. It is crucial to select appropriate network architectures for a given application, if the network is too small, it may not generalize the training data sufficiently well, on the other hand, if the network is too large, it may result in overfitting, thus producing poor generalization performance on unseen data.

Regarding to the problem of network architecture selection, some researchers

advocated that the network architecture should not be determined by a trial and error method but should be determined by learning algorithm[6]. In the context of ELM, some learning algorithms have been proposed to determine the architecture of SLFNs trained with ELM, these methods can be roughly classified into tow categories: incremental methods and pruning methods. The first incremental method is Incremental-ELM (I-ELM) proposed by Huang et al[7], the variants of I-ELM were proposed in [8] and [9]. Another famous incremental approach referred to as error minimized extreme learning machine (EM-ELM)[6] is proposed by Feng et al. All incremental approaches first start from a small SLFNs, and then randomly add hidden nodes one by one (or group by group) to the SLFNs until a predefined criterion is satisfied. The main drawback of the incremental approaches is that the optimal network architecture can not be obtained automatically[10]. Usually, the halt criterion of the incremental approaches is a predefined maximal number of hidden nodes or a predefined training accuracy. The pruning methods start from a large SLFNs and attempt to optimize the structure of the SLFNs by pruning the unnecessary nodes. The representative pruning methods include the Optimally Pruned ELM (OP-ELM)[11] and the Pruned-ELM (P-ELM)[12]. OP-ELM first ranks the hidden nodes by applying multi-response sparse regression algorithm (MRSR)[13] and then selects the nodes through LOO validation. There are two P-ELM algorithms: P-ELM1 and P-ELM2, which firstly uses the Chi-squared ($\chi^2$) and information gain (IG) respectively to measure the relevance between the hidden nodes and the class labels, and then both P-ELM1 and P-ELM2 employ a base set of relevance threshold values, i.e. $\gamma_i$, $i = 1, 2, \cdots, q$, rather than a single threshold to determine the architecture of SLFNs. For each $\gamma_i$, an architecture of SLFNs is obtained, the final network is selected from the $q$ networks using AIC measure[14]. The main drawback of P-ELM including P-ELM1 and P-ELM2 is the high computational complexity due to the discretization for estimating the probability density of continuous variables. In addition, the discretization will inevitably lead to information loss. In order to deal with this problem, based on tolerance rough sets, this paper proposed an improved pruned-ELM algorithm named TRS-PELM, which can overcome the drawbacks mentioned above.

TRS-PELM employs the dependency degree in tolerance rough set to measure the relevance between the hidden nodes and the class labels. As does in P-ELM, the final network architecture is also determined with a base set of potential relevance threshold values using the $C_p$ statistic[15] rather than AIC statistic, because that the information of full model as reference model is integrated into $C_p$, and it has a sound mathematical foundation. The theoretical analysis and experimental study of the proposed method on 8 benchmark classification applications show that the proposed approach leads to a compact architecture which generates the comparable generalization performance.

This paper is organized as follows. In Section 2, the ELM and the tolerance rough set are briefly reviewed, Criteria for architecture selection are discussed in Section 3, the proposed algorithm TRS-PELM and the computing complexity analysis are

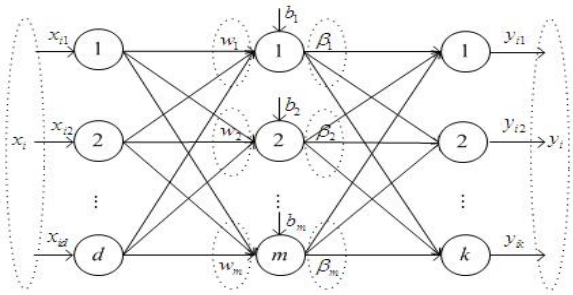

Fig. 1. The single-hidden layer feed-forward neural network

given in Section 4, and Section 5 presents the experimental results. Finally Section 6 concludes this paper.

## 2. Preliminaries

Extreme learning machine and tolerance rough set are briefly reviewed in this section.

### 2.1. *Extreme Learning Machine (ELM)*

The ELM algorithm was proposed by Huang et al. for training single-hidden layer feed-forward neural networks (SLFNs), see Figure 1. According to Theorem 2.1 of references [1, 3], the input weights and biases do not need to be adjusted, and it is possible to analytically determine the output weights by finding the least-square solution. The neural network is obtained after very few steps with very low computational cost.

Given a training data set, $D = \{(x_i, y_i)|x_i \in R^d, t_i \in R^k, i = 1, 2, \cdots, n\}$, where $x_i$ is a $d \times 1$ input vector and $t_i$ is a $k \times 1$ target vector, a SLFN with $m$ hidden nodes is formulated as

$$f(x_i) = \sum_{j=1}^{m} \beta_j g(w_j \cdot x_i + b_j),\ i = 1, 2, \cdots, n \tag{1}$$

where $w_j = (w_{j1}, w_{j2}, \cdots, w_{jd})^T$ is the weight vector connecting the $j$th hidden node with the input nodes. $b_j$ is the threshold of the $j$th hidden node. $w_j$ and $b_j$ are randomly assigned. $\beta_j = (\beta_{j1}, \beta_{j2}, \cdots, \beta_{jm})^T$ is the weight vector connecting the $j$th hidden node with the output nodes. The parameters $\beta_j (j = 1, 2, \cdots, m)$ may be estimated by least-square fitting with the given training data set $D$, i.e. , satisfying

$$f(x_i) = \sum_{j=1}^{m} \beta_j g(w_j \cdot x_i + b_j) = t_i \tag{2}$$

Equation (2) can be written in a more compact format as

$$H\beta = T \tag{3}$$

where

$$H = \begin{pmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_m \cdot x_1 + b_m) \\ \vdots & \cdots & \vdots \\ g(w_1 \cdot x_n + b_1) & \cdots & g(w_m \cdot x_n + b_m) \end{pmatrix} \tag{4}$$

$$\beta = (\beta_1^T, \cdots, \beta_m^T)^T \tag{5}$$

and

$$T = (t_1^T, \cdots, t_n^T)^T \tag{6}$$

$H$ is the hidden layer output matrix of the network[1,2], where the $j$th column of $H$ is the $j$th hidden nodes output vector with respect to inputs $x_1, x_2, \cdots, x_n$, and the $i$th row of $H$ is the output vector of the hidden layer with respect to input $x_i$. If the number of hidden nodes is equal to the number of distinct training samples, the matrix $H$ is square and invertible, and SLFNs can approximate these training samples with zero error. But generally, the number of hidden nodes is much less than the number of training samples. Therefore, $H$ is a non-square matrix and one cannot expect an exact solution of the system (3). Fortunately, it has been proved in [3] that SLFNs with random hidden nodes have the universal approximation capability and the hidden nodes could be randomly generated. The least-square fitting is to solve the following equation.

$$\min_{\beta} = \|H\beta - T\| \tag{7}$$

The smallest norm least-squares solution of (7) may be easily obtained:

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

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $H$.

The ELM Algorithm[1] is presented in the following.

**ELM Algorithm**

**Input:** Training data set $D = \{(x_i, y_i)|x_i \in R^d, t_i \in R^k, i = 1, 2, \cdots, n\}$, an activation function $g$, and the number of hidden nodes $m$.

**Output:** weights matrix $\beta$

**Step1.** Randomly assign input weights $w_j$ and $b_j$,$j = 1, 2, \cdots, m$;

**Step2.** Calculate the hidden layer output matrix $H$;

**Step3.** Calculate output weights matrix $\beta = H^{\dagger}T$.

### 2.2. *Tolerance Rough Set (TRS)*

Rough set (RS)[16] originally proposed by Pawlak in 1982 provides an effective and efficient tool to deal with vagueness and uncertainty. The one of appealing applications of RS is attribute reduction in decision table, which is a tuple $DT = (U, C \cup D)$, where $U$ is a set of objects named the universe of discourse, $C$ is the set of conditional attributes and $D$ is the set of decision attributes. The reduct is a collection of conditional attributes set $C$ that contain the same discriminating information.

The basis notion of RS is equivalence relation induced from a subset of conditional attributes set $C$ with descrete-value. The classical RS can only deal with decision table with descrete-valued conditional attributes. In 1996, Skowron and Stepaniuk proposed tolerance rough set (TRS)[17] which is an extension of classical RS, TRS can deal with real-valued conditional attributes. In TRS, the equivalence relation was replaced with the similarity relation. In the following, we will briefly review the concepts of TRS used in this paper.

**Definition 1.** Given a decision table $DT = (U, C \cup D)$, $R$ is a similarity relation on $U$ if the following three requirements hold:
(1) $R$ is a binary relation on $U$;
(2) $R$ is reflective, i.e. $R(x, x) = 1, \forall x \in U$;
(3) $R$ is symmetric, i.e. $R(x, y) = R(y, x), \forall x, y \in U$.

Given a decision table $DT = (U, C \cup D)$, one can define many similarity relations on $U$[18,19], such as:

$$R_a(x, y) = 1 - \frac{|a(x) - a(y)|}{|a_{max} - a_{min}|} \tag{9}$$

$$R_a(x, y) = exp\left(-\frac{(a(x) - a(y))^2}{2\sigma_a^2}\right) \tag{10}$$

$$R_a(x, y) = max\left(min\left(\frac{a(y) - (a(x) - \sigma_a)}{a(x) - (a(x) - \sigma_a)}, \frac{(a(x) + \sigma_a) - a(y)}{(a(x) + \sigma_a) - a(x)}, 0\right)\right) \tag{11}$$

where $a \in C$, $a(x)$ and $a(y)$ are the value of instance $x$ and $y$ in conditional attribute $a$ respectively. $a_{max}$ and $a_{min}$ are the maximum and minimum values of a on $U$, respectively. $\sigma_a$ is the variance of conditional attribute $a$.

**Definition 2.** Given a decision table $DT = (U, C \cup D)$, $P \subseteq C$ , the similarity relation induced from $P$ can be defined as follows:

$$R_{P,\tau}(x, y) = \frac{\sum_{a \in P} R_a(x, y)}{|P|} \geq \tau \tag{12}$$

or

$$R_{P,\tau}(x, y) = \prod_{a \in P} R_a(x, y) \geq \tau \tag{13}$$

where $\tau \in [0, 1]$ is threshold of similarity.

**Definition 3.** Given a decision table $DT = (U, C \cup D)$, $x \in U$, $P \subseteq C$, $R_P$ is a similarity relation induced from $P$, the $\tau$ similarity class or $\tau$ tolerance class of $x$ is defined as follows.

$$[x]_{R_{P,\tau}} = \{y | (y \in U) \wedge (xR_{P,\tau}y)\} \tag{14}$$

**Definition 4.** Given a decision table $DT = (U, C \cup D)$, $X \subseteq U$, the $\tau$ tolerance lower approximation and $\tau$ tolerance upper approximation of $X$ is defined as follows:

$$\underline{R}_{P,\tau}(X) = \{x|(x \in U) \wedge ([x]_{R_{P,\tau}} \subseteq X)\} \tag{15}$$

and

$$\overline{R}_{P,\tau}(X) = \{x|(x \in U) \wedge ([x]_{R_{P,\tau}} \cap X \neq \phi)\} \tag{16}$$

**Definition 5.** Given a decision table $DT = (U, C \cup D)$, $P \subseteq C$, $R_P$ is a similarity relation induced from $P$, the $\tau$ relative positive region of $D$ with respect to $P$ is defined as follows:

$$POS_{P,\tau}(D) = \bigcup_{U_i \in U/D} \underline{R}_{P,\tau}(U_i) \tag{17}$$

**Definition 6.** Given a decision table $DT = (U, C \cup D)$, $P \subseteq C$, $R_P$ is a similarity relation induced from $P$, the $\tau$ tolerance dependency of $D$ with respect to $P$ is defined as follows:

$$\gamma_{P,\tau}(D) = \frac{|POS_{R_{p,\tau}}(D)|}{|U|} \tag{18}$$

In this paper, TRS-PELM uses the $\tau$ tolerance dependency of $D$ with respect to $P$ to assess the importance of the hidden nodes.

## 3. Criteria for Architecture Selection

In the framework of ELM, the architecture selection is to determine the optimal number of the hidden nodes. Because that after the nonlinear random mappings of the SLFN are fixed, the network will correspond to a linear regression model (without intercept). Hence the ELM networks architecture selection can be viewed as model selection or variable selection in linear regression. The criteria for choice of best model in linear regression can be borrowed to select the optimal architecture of ELM networks. For the purpose of comparison and analysis, three commonly used criteria are presented, which are *PRESS* statistic[20], *AIC*[14] and $C_p$ statistic[15].

**(1) *PRESS* statistic**

For the given data set $D$, set aside the $i$th observation $x_i$ from $D$, and use the remaining $n$-1 observations to estimate the coefficients for a particular candidate model. Each observation is removed one at a time, resulting in $n$ prediction errors named *PRESS* residuals $e_{i,-i} = y_i - \hat{y}_{i,-i}, (i = 1, 2, \cdots, n)$. These *PRESS* residuals are true prediction errors with $\hat{y}_{i,-i}$ being independent of $y_i$. Hence, *PRESS* (PREdiction Sum of Square) statistic is a cross-validation based criterion. *PRESS* is defined as

$$PRESS = \sum_{i=1}^{n} (e_{i,-i})^2 = \sum_{i=1}^{n} (y_i - \hat{y}_{i,-i})^2 \tag{19}$$

So for choice of best model, one might favor the model with the smallest *PRESS*. *PRESS* can be computed efficiently using the following formula and does not require repeated regressions[20].

$$e_{i,-i} = \frac{y_i - \hat{y}_i}{1 - x_i^T (X^T X)^{-1} x_i} = \frac{e_i}{1 - h_{ii}} \tag{20}$$

then

$$PRESS = \sum_{i=1}^{n} (\frac{e_i}{1 - h_{ii}})^2 \tag{21}$$

In OP-ELM, after ranking the hidden nodes by LARS, the optimal number of hidden nodes is selected by *PRESS*. Although *PRESS* can be computed efficiently with (21), it is also time consuming due to the computation of $(X^T X)^{-1}$, the computational complexity is $O(m^3)$[20], where $m$ is the order of the data matrix $X$. Because that pruning algorithms start from a big network, the order of the data matrix $X$ is usually very high, i.e. $m$ is usually a big number. The total computational complexity for computing *PRESS* is $O(m^3 + n)$, where $n$ is the number of samples.

**(2)*AIC***

*AIC* (Akaike Information Criterion) is a criterion derived from information theory by Akaike in 1973[14]. It has the following general form:

$$AIC = n \times log - likelihood + 2 \times number\,of\,parameters \tag{22}$$

For normal distribution models, it becomes,

$$AIC = n \times ln(SSE) + 2p \tag{23}$$

where $SSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$.

One favors the candidate model with the smallest *AIC* value. Obviously, the computational complexity for computing *AIC* is $O(n)$. P-ELM employed *AIC* to evaluate $q$ candidate models and then select the optimal one which has the smallest *AIC* value.

**(3)$C_p$ statistic**

The $C_p$ statistic was proposed by Mallow[15]. Assume that the model include $p$ variables, the $C_p$ statistic is defined as

$$C_p = \frac{SSE_p}{\sigma^2} - n + 2p \tag{24}$$

where $\sigma^2$ is the variance of residuals. For real problem, $\sigma^2$ is unknown and has to be estimated. Usually one use $\hat{\sigma}^2 = \frac{1}{n-m-1} SSE_m$ to estimate $\sigma^2$, where $m$ is number of variables of full model. The computational complexity for computing $C_p$ is also $O(n)$. The computational complexity for computing $C_p$ and *AIC* is same, but the information of full model as reference model is integrated into $C_p$, and it

has a sound conceptual base[21]. Hence, as criterion for choosing the best model, comparing with $AIC, C_p$ is more appropriate. In this paper, $C_p$ is employed to evaluate the candidate models and the optimal network architecture is determined when $C_p$ reaches its minimum.

## 4. The proposed method TRS-PELM and the analysis of computational time complexity

In this section, the proposed algorithm TRS-PELM is firstly presented in 4.1. In order to show the effectiveness of the TRS-PELM, a theoretical analysis of the computational time complexity of TRS-PELM is given in 4.2, the analysis of the computational time complexity of P-ELM is also given for comparison in this section.

### 4.1. *The proposed method TRS-PELM*

The main idea of the proposed TRS-PELM lies in identifying the degree of relevance (i.e. the dependency degree) between the hidden nodes and the class labels with the tolerance rough set and then pruning the irrelevant or low relevant hidden nodes from the SLFN. As does in P-ELM, the final network architecture and the generalization ability are systematically determined from a base set of relevance threshold values $\lambda_i (i = 1, 2, \cdots, q)$ using the $C_p$ statistic rather than $AIC$ statistic. The proposed algorithm TRS-PELM include three phases: initialization phase, pruning phase and testing phase, the proposed algorithm TRS-PELM is described in detail as follows:

**TRS-PELM Algorithm**

**Input:** A data set $DT = \{(x_i, y_i)|x_i \in R^d, t_i \in R^k, i = 1, 2, \cdots, n\}$, activation function $g$, an initial large hidden node size $m$, and a relevance threshold base $\lambda(\lambda_1, \lambda_2, \cdots, \lambda_q)$.

**Output:** An optimal SLFN

**Phase 1 Initialization phase:**

Randomly partition the data set $DT$ into three non-overlapping subsets: training set $DT_1$, validation set $DT_2$ and testing set $DT_3$;

Randomly assign hidden node parameters $(w_i, b_i) i = 1, \ldots, m$ ;

Train the initialized SLFN with training set $DT_1$ using ELM algorithm, obtained the hidden layer output matrix $H$.

**Phase 2 Pruning phase:**

Calculate the dependency degree with the validation set $DT_2$ using equation (18) and sort them in descending order;

For each relevance threshold $\lambda_i (i = 1, 2, \cdots, q)$ , prune the nodes with relevance value less than $\lambda_i$ from the trained SLFN, obtain the pruned SLFN denoted by $SLFN_{(i)} (i = 1, 2, \cdots, q)$;

For each pruned $SLFN_{(i)} (i = 1, 2, \cdots, q)$, calculate its $C_p$ value denoted by $C_p(i) (i = 1, 2, \cdots, q)$ with (24);

Calculate $i^* = argmin_{1 \le i \le q} C_p(i)$;

Output $SLFN_{(i^*)}$.

**Phase 3 Testing phase:**

Retrain the network $SLFN_{(i^*)}$ obtained in phase 2 with the training data $DT_1$ using ELM algorithm;

Evaluate the performance on the testing data set $DT_3$.

### 4.2. *The analysis of computational time complexity*

In the following, a theoretical analysis of the computational time complexity of the proposed algorithm TRS-PELM and P-ELM are presented to verify the effectiveness of TRS-PELM.

The main steps of TRS-PELM and P-ELM are same, which are: (1) Partition the data set into three subsets: training set, validation set and testing set; (2) SLFN construction using ELM, (3) Ranking of the hidden nodes; (4) Selection of the optimal SLFN; (5) Retrain and test the selected SLFN with training set and testing set. the main difference between TRS-PELM and P-ELM are embedded in step (3) and (4). In step (3) P-ELM use $\chi^2$ and information gain to measure the significance of individual hidden node, while TRS-PELM employs the dependency degree to measure the significance of individual hidden node. In step (4), P-ELM use $AIC$ statistic to select the optimal SLFN, while TRS-PELM employs $C_p$ statistic to select the optimal SLFN.

We first analyze the the computational time complexity of TRS-PELM. The computational time complexity of step (1) of TRS-PELM is $O(n)$. The computational time complexity of step (2) of TRS-PELM is the one of ELM. It is well known that the main computational cost of ELM comes from the calculation of the Moore-Penrose generalized inverse of hidden layer output matrix $H$. Huang et al.[1] pointed out that, when the $n$ training samples are distinct, the hidden-layer output matrix $H$ is column full rank with probability one, so the ELM can be solved as a full-rank least-square problem[22]. For such a problem, some methods, such as orthogonal project, Householder triangularization, and Gram-Schmidt orthogonalization, may be used to solve it, with computational time complexity $O(m^2n)$[23]. Hence, the computational time complexity of ELM algorithm is $O(m^2n)$. When $m \ll n$, the computational time complexity of ELM can be thought to approach $O(n)$[22]. The step (3) of TRS-PELM consist of calculating the equivalence class and ordering the attributes by the dependency degree, the computational time complexity are $O(mn^2)$ and $O(mlog_2m)$, accordingly, the computational time complexity of step (3) of TRS-PELM is $O(mn^2 + mlog_2m)$. As indicated in section 3, the computational time complexity of step (4) of TRS-PELM is $O(qn)$ introduced by computing $C_p$ statistic, the computational time complexity of step (5) of TRS-PELM is $O(m^2n)$. Finally, the computational time complexity of TRS-PELM is $O(n)+O(m^2n)+O(mn^2 + mlog_2m)+O(qn)+O(m^2n)$. Generally, the cardinality of relevance threshold base $q$ is far less than the number of samples $n$. So the com-

putational time complexity of TRS-PELM is $O(mn^2) + O(+mlog_2m)$ in the worst situation.

Apparently, the computational time complexity of step (1) and (2) of P-ELM are $O(n)$ and $O(m^2n)$ respectively. The main computational cost of P-ELM comes from stage (3) in which the probability distribution of the variables must be estimated for calculating $\chi^2$ and IG. In P-ELM, the entropy based method of discretization of variables is employed to estimate the probability distribution of the variables. It is well known that the computational time complexity of the entropy based discretization algorithm is $O(mn^2 + mnlog_2n)$[24]. Also as indicated in section 3, the computational time complexity of stage (4) of P-ELM is $O(qn)$ introduced by computing $AIC$ statistic, the computational time complexity of stage (5) of P-ELM is $O(m^2n)$. Hence, the computational time complexity of P-ELM is $O(n)+O(m^2n)+O(mn^2 + mnlog_2n) + O(qn) + O(m^2n)$. In the worst situation, the computational time complexity of P-ELM is $O(mn^2 + mnlog_2n))$.

Based on the above analysis, it can be seen that the computational time complexity of TRS-ELM is less than the one of P-ELM, in the following section, the results will be confirmed with experiments.

## 5. Experimental Results and Analysis

The proposed algorithm TRS-PELM is tested on the 8 UCI data sets[25] by 10-fold cross-validation in the environment of Matlab 7.0 on a Pentium 4 PC. For each data set, the 10-fold cross-validation are run 10 times. The experimental results are the average of the 10 outputs. The basic information of the selected data sets is shown in table 1. All the input data are firstly normalized into [-1, 1], and the number of hidden nodes used in ELM is chosen by a trial and error method. Three experiments are conducted to verify the effectiveness of the proposed method. The parameter selection is presented in experiment 1. The feasibility analyses of the proposed algorithm TRS-PELM are presented in experiment 2. The comparison between the proposed method with P-ELM is conducted in experiment 3.

As advice in [26], the partition of data sets employed in all 2 experiments is 50% for training, and 25% each for validation and testing. For the fair comparison with other related methods, as used in [12], the relevance threshold range is also in the range of [0.1, 0.9], the step size is also 0.1. With respect to the number of hidden nodes, according to theorem 2.2 of [1], the upper bound of the required number of hidden nodes is the number of distinct training samples, i.e. $m \leq n$. Therefore, there is a thumb rule that the number of hidden nodes is initialized as equal to the size of the training data as considered in the present study[12]. Nevertheless for large data sets, this may be impractical, and it is should be highlight that a large initial hidden nodes will result in increasing computational burden.

Table 1. The basic information of the 8 data sets used in the experiments

| Data sets | #Attributes | #Classes | #Samples |
|---|---|---|---|
| Iris | 4 | 3 | 150 |
| Glass | 10 | 7 | 214 |
| Ionosphere | 34 | 2 | 351 |
| Statlog | 14 | 2 | 690 |
| Diabetes | 8 | 2 | 768 |
| Vehicle | 18 | 4 | 846 |
| Contraceptive | 9 | 3 | 1473 |
| Car | 6 | 4 | 1728 |

### 5.1. *Experiment 1: Parameter selection*

Because the choice of the relevance threshold directly impacts the architecture and generalization performance of the selected network, we select the relevance threshold as opposed to a trial and error process. Here, we only use the experimental results on vehicle data set as example to illustrate the effects of the relevance threshold on the validation accuracy of TRS-PELM. Figure 2 shows the relationship of validation accuracy and the selected relevance threshold. A relevance threshold step size of 0.05 is considered for investigating its influence on validation accuracy. The threshold value of $\lambda$ defines that the hidden nodes with the threshold value which is less than $\lambda$ are pruned. Further, the threshold value of 0 implies that no hidden nodes are pruned and the 1 implies that all the hidden nodes are pruned. From Figure 2, it is observed that an initial large size node would result poor performance on unseen samples, due to the effects of overfitting on the training data. The Figure 2 illustrates that with increasing relevance threshold values, the validation accuracy also improves, but, when it is larger than one value, the validation accuracy becomes to degrease. This is because of underfitting. From the results in Figure 2 (similar results can be are observed in other 7 datasets), the range of [0, 1] is effective for generating compact networks that generates competitive prediction accuracy. Hence, we suggest using a relevance threshold value $\lambda$ which is in the interval of [0, 1] and at a step size of 0.05.

With different number of initial hidden nodes, the number of the selected nodes and the testing accuracy are presented in Figure 3 and Figure 4. For the dataset vehicle, the number of selected nodes and the testing accuracy is stable when the initial hidden nodes is above some value, so, it is better to choose a small value of the initial nodes that leads to a compact network structure. The similar observations can be obtained on other 7 data sets. In our simulations, for the small data sets, the number of initial hidden nodes is set to be the number of the training data, but, when the number of the data set is large than 1000, it is set to be 500.

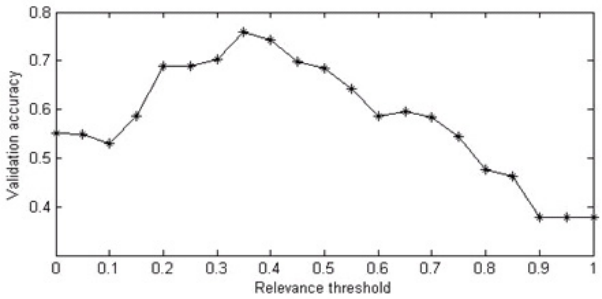


Fig. 2. Effect of the relevance threshold on the validation accuracy on the vehicle application

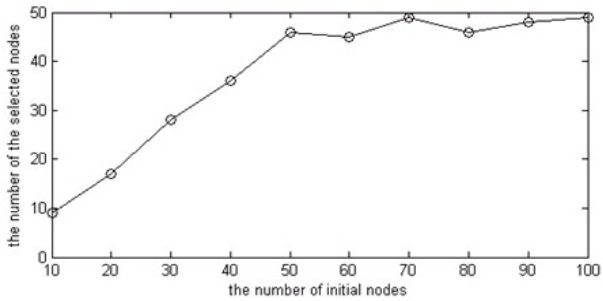


Fig. 3. Effect of the different initial hidden nodes on the selected nodes on the vehicle application

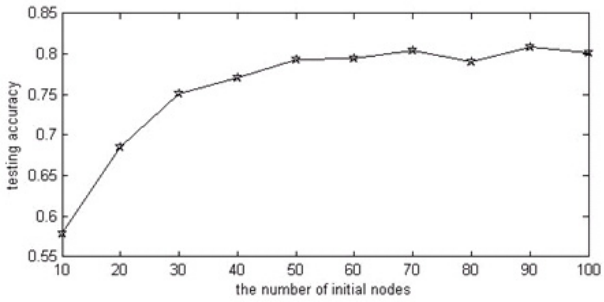


Fig. 4. Effect of the different initial hidden nodes on the testing accuracy on the vehicle application

### 5.2. *Experiment 2: the feasibility analysis of the proposed algorithm TRS-PELM*

In this experiment, the proposed algorithm TRS-PELM is compared with ELM and BP algorithm to show the feasibility of TRS-PELM. The performance of the proposed algorithm TRS-PELM is presented in terms of testing accuracy on unseen data, CPU time and selected hidden node size. Simulation results for eight data sets

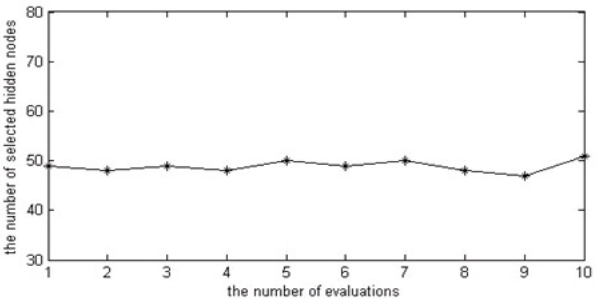


Fig. 5. The number of selected hidden nodes on vehicle dataset in different evaluations

are presented in table 2. In comparison to the ELM, it is clear that the proposed TRS-PELM arrives at competitive testing accuracy, but much compact network structure and lower training time. Note that in ELM, the training time involves a trial-and-error process to identify a suitable network structure. In contrast to BP, the proposed TRS-PELM needs less training time. BP need to tune iteratively, so it spends more time. Furthermore, the TRS-PELM arrives at higher prediction accuracy on unseen data besides Glass and Contraceptive cases. The number of selected hidden nodes on vehicle data set in different evaluations is given in Figure 5, which illustrates that the TRS-PELM is stable and guarantee the excellent robustness in the generalization ability of the network. Accordingly, comparing with ELM and BP algorithm, the proposed algorithm TRS-PELM is feasible.

### 5.3. *Experiment 3: the comparison between the proposed method with P-ELM*

In this section, the experimental comparisons on performance including the testing accuracy and the number of selected hidden nodes between TRS-PELM and P-ELM are presented.

The proposed algorithm TRS-PELM is compared with P-ELM in two aspects: the testing accuracy and the number of selected hidden nodes. P-ELM includes P-ELM 1 and P-ELM2; P-ELM1 uses $\chi^2$ to prune the hidden nodes, while P-ELM2 uses IG to prune the hidden nodes. The comparison results are listed in Table 3. For each data set, the experiment is run 10 times. The experimental results are the average of the 10 outputs. Comparing with P-ELM1 and P-ELM2, in eight data sets, there is only one data set Contraceptive on which the testing accuracies of the TRS-ELM is lower than the ones of P-ELM1 and P-ELM2. With respect to the number of selected hidden nodes, the number determined by TRS-PELM is the smallest one. In other works, TRS-PELM can produce significantly more compact networks architecture while keeping the generalization ability compared with P-ELM1 and P-ELM2.

Table 2. The performance comparison between the TRS-PELM, ELM and BP algorithm

| Data sets | Algorithms | #Selected nodes | Testing accuracy(%) | CPU time(s) |
|---|---|---|---|---|
| Iris | TRS-PELM | 8.0 | 97.33 | 0.3641 |
| | ELM | 18.0 | 96.00 | 0.9688 |
| | BP | 6.0 | 96.93 | 14.2188 |
| Glass | TRS-PELM | 14.0 | 76.44 | 0.6859 |
| | ELM | 16.0 | 75.33 | 25.5469 |
| | BP | 5.0 | 79.94 | 17.4688 |
| Ionosphere | TRS-PELM | 23.7 | 90.34 | 1.0703 |
| | ELM | 153.0 | 72.59 | 12.3594 |
| | BP | 8.0 | 86.23 | 20.7813 |
| Statlog | TRS-PELM | 20.3 | 84.64 | 2.0000 |
| | ELM | 35.0 | 83.77 | 8.5313 |
| | BP | 9.0 | 84.10 | 22.1875 |
| Diabetes | TRS-PELM | 8.4 | 78.39 | 2.1578 |
| | ELM | 24.0 | 76.82 | 8.5625 |
| | BP | 8.0 | 76.18 | 22.9531 |
| Vehicle | TRS-PELM | 29.5 | 80.48 | 2.7578 |
| | ELM | 48.0 | 79.91 | 22.5625 |
| | BP | 18.0 | 79.87 | 31.8125 |
| Contraceptive | TRS-PELM | 11.0 | 50.07 | 5.5422 |
| | ELM | 23.0 | 50.73 | 48.0156 |
| | BP | 10.0 | 53.93 | 30.0781 |
| Car | TRS-PELM | 95.2 | 89.55 | 27.2797 |
| | ELM | 178.0 | 89.24 | 137.1250 |
| | BP | 50.0 | 87.84 | 103.9375 |

## 6. Conclusions

This paper attempts to design the optimal network structure of ELM classifier by applying a pruned algorithm named TRS-PELM. TRS-PELM cleverly combines tolerance rough set and ELM to obtain a compact network structure. The proposed algorithm TRS-PELM employs the dependency degree to measure the significance of hidden nodes. The hidden nodes with dependency degree value less than a threshold will be pruned from the SLFN. The final network architecture and the generalization ability are systematically determined from a base set of potential relevance threshold values using the $C_p$ statistic. The experimental comparison of performance of the

Table 3. The performance comparison between the TRS-PELM, P-ELM1 and P-ELM2

| Data sets | Algorithms | #Selected nodes | Testing accuracy(%) |
|---|---|---|---|
| Iris | TRS-PELM | 8.00 | 97.33 |
| | P-ELM1 | 13.44 | 96.23 |
| | P-ELM2 | 14.71 | 97.18 |
| Glass | TRS-PELM | 14.00 | 76.44 |
| | P-ELM1 | 18.07 | 65.14 |
| | P-ELM2 | 20.05 | 65.08 |
| Ionosphere | TRS-PELM | 23.70 | 90.34 |
| | P-ELM1 | 41.14 | 89.22 |
| | P-ELM2 | 42.00 | 88.38 |
| Statlog | TRS-PELM | 20.30 | 84.64 |
| | P-ELM1 | 37.21 | 82.56 |
| | P-ELM2 | 39.00 | 84.14 |
| Diabetes | TRS-PELM | 8.40 | 78.39 |
| | P-ELM1 | 33.15 | 75.24 |
| | P-ELM2 | 38.00 | 78.16 |
| Vehicle | TRS-PELM | 29.50 | 80.48 |
| | P-ELM1 | 67.24 | 79.25 |
| | P-ELM2 | 74.18 | 79.14 |
| Contraceptive | TRS-PELM | 11.00 | 50.07 |
| | P-ELM1 | 42.05 | 58.04 |
| | P-ELM2 | 46.41 | 58.00 |
| Car | TRS-PELM | 95.20 | 89.55 |
| | P-ELM1 | 236.15 | 87.07 |
| | P-ELM2 | 240.38 | 86.14 |

proposed algorithm TRS-PELM with other well-known related algorithms including the ELM, BP algorithm and P-ELM are carried out on 8 benchmark data sets. The theoretical analysis of computational time complexity and the experimental results demonstrate that the proposed algorithm TRS-PELM can effectively achieve more compact network architecture with competitive testing accuracy.

**Acknowledgements**

This research is supported by the national natural science foundation of China (61170040, 71371063), by the natural science foundation of Hebei Province

(F2013201110, F2013201220), by the Key Scientific Research Foundation of Education Department of Hebei Province (ZD20131028).

## References

1. G. B. Huang, Q. Y. Zhu, C. K. Siew. "Extreme learning machine: Theory and applications", *Neurocomputing*. **70**(2006)489-501.
2. G. B. Huang, D. H. Wang, Y. Lan. "Extreme learning machines: a survey", *International Journal of Machine Learning and Cybernetics*. **2**(2011)107-122.
3. G. B. Huang, L. Chen, C. K. Siew. "Universal approximation using incremental constructive feedforward networks with random hidden nodes", *IEEE Transactions on Neural Networks*. **17**(2006)879-892.
4. S. Haykin. "Neural Networks: A Comprehensive Foundation", (Prentice Hall, New Jersey, 1999).
5. C. Cortes, V. Vapnik. "Support vector networks", *Machine Learning*. **20**(1995)273-297.
6. G. Feng, G. B. Huang, Q. Lin, R. Gay. "Error minimized extreme learning machine with growth of hidden nodes and incremental learning", *IEEE Transactions on Neural Networks*. **20**(2009)1352-1357.
7. G. B. Huang, M. B. Li, L. Chen, C. K. Siew. "Incremental extreme learning machine with fully complex hidden nodes", *Neurocomputing*. **71**(2008)576-583.
8. G. B. Huang, L. Chen. "Enhanced random search based incremental extreme learning machine", *Neurocomputing*. **71**(2008)3060-3068.
9. G. B. Huang, L. Chen. "Convex incremental extreme learning machine", *Neurocomputing*. **70**(2007)3056-3062.
10. Y. Lan, Y. C. Soh, G. B. Huang. "Constructive hidden nodes selection of extreme learning machine for regression", *Neurocomputing*. **73**(2010)3191-3199.
11. Y. Miche, A. Sorjamaa, P. Bas, O. Simula. "OP-ELM: Optimally pruned extreme learning mMachine", *IEEE Transactions on Neural Networks*. **21**(2010)158-162.
12. H. J. Rong, Y. S. Ong, A. H. Tan, Z. Zhu. "A fast pruned-extreme learning machine for classification problem", *Neurocomputing*. **72**(2008)359-366.
13. T. Similä, J. Tikka. "Multiresponse sparse regression with application to multidimensional scaling", *Proceeding of International Conference on Artificial Neural Networks*. **3697/2005**(2005)97-102.
14. H. Akaike. "Information theory and an extension of the maximum likelihood principle", In B. N. Petrov and F. Csaki (Eds.), *Second international symposium on information theory*.Budapest: Academiai Kiado(1973)267-281.
15. C. L. Mallows. "Some comments on $C_p$", *Technometrics*, **42**(2000)87-94.
16. Z. Pawlak. "Rough sets". *International Journal of Computer and Information Sciences*. **11**(1982)341-356.
17. A. Skowron, J. Stepaniuk. "Tolerance Approximation Spaces". *Fundamenta Informaticae*. **27**(1996)245-253.
18. N. Parthalain, Q. Shen, R. Jensen. "A distance measure approach to exploring the rough set boundary region for attribute reduction". *IEEE Transactions on Knowledge and Data Engineering*. **22**(2010)305-317.
19. N. Parthalain, Q. Shen. "Exploring the boundary region of tolerance rough sets for feature selection", *Pattern Recognition*. **42**(2009)655-667.
20. R. H. Myers. "Classical and modern regression with applications", Second Edition, (Pacific Grove, CA: Duxbury, 1990).
21. X. Yan, X. G. Su. "Linear regression analysis: theory and computing", (World Scien-

tific Publishing Co. Pte. Ltd., 2009).
22. X. Liu, C. Gao, P. Li. "A comparative analysis of support vector machines and extreme learning machines", *Neural Networks.* **33**(2012)58-66.
23. G. Golub, F. Charles, V. Loan. "Matrix computations", (Baltimore: Johns Hopkins University Press,1983).
24. J. Dougherty, R. Kohavi, M. Sahami. "Supervised and unsupervised discretization of continuous features", *Proceedings of the Twelfth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA,(1995), 194-202.
25. A. Frank, A. Asuncion. "UCI machine learning repository", Irvine, CA: University of California, School of Information and Computer Science, [http://archive.ics.uci.edu/ml], 2010.
26. T. Hastie, R. Tibshirani, J. Friedman. "The elements of statistical learning: data mining, inference, and prediction", Second Edition, (New York: Springer-Verlag, 2009).