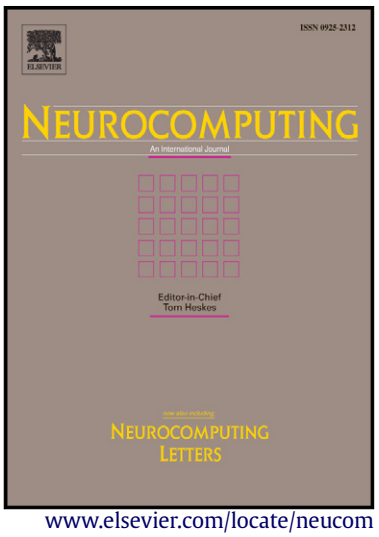

www.elsevier.com/locate/neucom

Ensemble delta test-extreme learning machine (DT-ELM) for regression

Qi Yu, Mark van Heeswijk, Yoan Miche, Rui Nian, Bo He, Eric Séverin, Amaury Lendasse

PII: S0925-2312(13)00975-2
DOI: http://dx.doi.org/10.1016/j.neucom.2013.08.041
Reference: NEUCOM13699

To appear in: *Neurocomputing*

Received date: 20 June 2013
Revised date: 19 August 2013
Accepted date: 26 August 2013

Cite this article as: Qi Yu, Mark van Heeswijk, Yoan Miche, Rui Nian, Bo He, Eric Séverin, Amaury Lendasse, Ensemble delta test-extreme learning machine (DT-ELM) for regression, *Neurocomputing,* http://dx.doi.org/10.1016/j.neucom.2013.08.041

Ensemble Delta Test- Extreme Learning Machine (DT-ELM) For Regression

Qi Yu*[a], Mark van Heeswijk[a], Yoan Miche[a], Rui Nian[e], Bo He[e], Eric Séverin[b], Amaury Lendasse[a,c,d]

[a] *Department of Information and Computer Science, Aalto University, FI-00076, Espoo, Finland*
[b] *LEM, Université Lille 1, 59043, Lille cedex, France*
[c] *IKERBASQUE, Basque Foundation for Science, 48011, Bilbao, Spain*
[d] *Computational Intelligence Group, Computer Science Faculty, University Of The Basque Country, Paseo Manuel Lardizabal 1, Donostia-San Sebastián, Spain*
[e] *College of Information and Engineering, Ocean University of China, 266003, Qingdao, China*

Abstract

Extreme learning machine (ELM) has shown its good performance in regression applications with a very fast speed. But there is still a difficulty to compromise between better generalization performance and smaller complexity of the ELM (number of hidden nodes). This paper proposes a method called Delta Test-ELM (DT-ELM), which operates in an incremental way to create less complex ELM structures and determines the number of hidden nodes automatically. It uses Bayesian Information Criterion (BIC) as well as Delta Test (DT) to restrict the search as well as to consider the size of the network and prevent overfitting. Moreover, ensemble modeling is used on different DT-ELM models and it shows good test results in Experiments Section.

Keywords: Extreme learning machine, Incremental Learning, Bayesian information criterion, Delta Test, Ensemble modeling.

1. Introduction

Extreme learning machine (ELM), which is a simple and efficient learning algorithm for single-hidden layer feedforward neural networks (SLFNs), has been recently proposed in [1]. ELM has shown good generalization performances for many real applications [2, 3, 4, 5] with an extremely fast learning speed [6, 7, 8, 9, 10, 11, 12]. However, like other similar approaches based on feedforward neural networks, some issues with the practical applications of the ELM still arise, most importantly, how to obtain the most appropriate architecture of the

Email address: qi.yu@aalto.fi (Qi Yu*)

network. In other words, how to select or search for the optimal number of hidden neurons remains a difficult problem.

In ELM, the generalization error is determined by changing the number of hidden neurons. Regardless of the fact that it is tedious and time consuming, it may bring two other severe problems: overfitting and the high complexity of the model, and it gets even serious when the data set presents high dimensionality and large number of observations.

Many methods have been introduced recently trying to choose the most suitable network structure of ELM and to further reduce the number of neurons without affecting the generalization performance. Pruning methods are one type of algorithms to address this problem. For example, Rong *et al.* in [13] proposed a pruned ELM (P-ELM) for classification, and Miche *et al.* in [11, 14] presented a method called optimally pruned ELM (OP-ELM). But pruning methods in general are rather inefficient since most of the time they are dealing with a network structure larger than necessary. On the other hand, some researchers manage to solve the problems via incremental learning. Like the Incremental extreme learning machine (I-ELM) [15] which adds the randomly generated hidden node one-by-one to the hidden layer until achieving an expected training accuracy or reaching the maximum number of hidden nodes. There are also some modifications made to I-ELM, like shown in [16, 17, 18]. However, these methods need to set the expected training error or maximum number of neurons in advance.

In this paper, a method called Delta Test - Extreme Learning Machine (DT-ELM) is proposed. It is operated in an incremental way but stops automatically if the Delta Test error remains unchanged for a certain number of iterations (this is the only parameter of the proposed method). . Bayesian Information Criterion (BIC) and Delta Test (DT) are used to minimize the Mean Square Error and find the suitable hidden layer neurons while avoiding overfitting. Moreover, ensemble modeling is performed on different DT-ELM models and it shows better test results.

2. Extreme Learning Machine

The Extreme Learning Machine algorithm is proposed by Huang *et al.* in [1] as an original way of building a single Hidden Layer Feedforward Neural Network (SLFN). The essence of ELM is that the hidden layer needs not to be iteratively tuned [1], and moreover, the training error $\parallel \mathbf{H}\beta - \mathbf{y} \parallel$ and the norm of the weights $\parallel \beta \parallel$ are minimized.

Given a set of N observations $(x_i, y_i), i \leq N.$ with $x_i \in \mathbf{R}^p$ and $y_i \in \mathbf{R}$. A SLFN with m hidden neurons in the hidden layer can be expressed by the following sum:

$$\sum_{i=1}^{m} \beta_i f(\omega_i x_j + b_i), \quad 1 \leq j \leq N \tag{1}$$

where β_i are the output weights, f be an activation function, ω_i the input weights and b_i the biases. Suppose the model perfectly describe the data, the

relation can be written in matrix form as $\mathbf{H}\beta = \mathbf{y}$, with

$$\mathbf{H} = \begin{pmatrix} f(\omega_1 x_1 + b_1) & \dots & f(\omega_m x_1 + b_m) \\ \vdots & \ddots & \vdots \\ f(\omega_1 x_n + b_1) & \dots & f(\omega_m x_n + b_m) \end{pmatrix} \tag{2}$$

$\beta = (\beta_1, ..., \beta_m)^T$ and $\mathbf{y} = (y_1, ..., y_n)^T$. The ELM approach is thus to initialize randomly the ω_i and b_i and compute the output weights $\beta = \mathbf{H}^\dagger \mathbf{y}$ by a Moore-Penrose pseudo-inverse [19].

The significant advantages of ELM are its extreme fast learning speed, and its good generalization performance while being a simple method [1]. There has been recent advances based on the ELM algorithm, to improve its robustness (OP-ELM [14], TROP-ELM [11], CS-ELM [20]), or make it a batch algorithm, improving at each iteration (EM-ELM [21], EEM-ELM [22]).

Along with the increase of number of hidden nodes in ELM, the error usually decreases. However, it also brings some corresponding difficulties: the complexity of the model and the overfitting problem. That's why Bayesian information criterion (BIC) (and Delta Test after it) is used in the proposed DT-ELM.

3. Bayesian Information Criterion and Delta Test

In the methodology presented in this paper, the Bayesian Information Criterion (BIC) and Delta Test (DT) are used as criteria in a "cascade" manner so as to select an appropriate number of neurons for the ELM structure and avoid overfitting. The following subsection detail the two criterion separately.

3.1. Bayesian information criterion (BIC)

The Bayesian information criterion (BIC) is one of the most widely known and pervasively used tools in statistical model selection, also known as Schwarzs information criterion (SIC) [23, 24]. It is based, in part, on the likelihood function, and it is closely related to Akaike information criterion (AIC)[25].

When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. The BIC resolves this problem by introducing a penalty term for the number of parameters in the model.

In brief, BIC is defined as:

$$\mathrm{BIC} = -2 \cdot \ln L + m \ln(N) \tag{3}$$

where,

- N–the number of observations, or equivalently, the sample size;
- m–the number of degrees of freedom remaining after fitting the model (free parameters to be estimated), with smaller value representing the better fits. If the estimated model is a linear regression, m is the number of parameters;

- L–the maximized value of the likelihood function for the estimated model.

Under some assumptions of model errors, BIC becomes the following formula for practical calculations [26]:

$$\mathrm{BIC} = N \cdot \ln(\widehat{\sigma_e^2}) + m \cdot \ln(N) \tag{4}$$

where $\widehat{\sigma_e^2}$ is the error variance.

Because BIC includes an adjustment for sample size, the BIC often favors a simpler model. In this paper, BIC is used to selected neurons incrementally in ELM, which are randomly generated and tested cluster by cluster. Therefore, BIC is calculated like:

$$\mathrm{BIC} = N \cdot \ln(MSE) + m \cdot \ln(N) \tag{5}$$

where N continues to be the number of samples, MSE represents the Mean Square error for the regression problem, and m is the number of neurons used in current model.

However, BIC is in theory designed only for data set of an infinite sample size, and in practice, it is really difficult to find the balance point between smaller error and not overfitting. Thus, using only BIC as a criterion to decide on an optimal number of neurons proved insufficient, and sometimes unreliable (especially in cases where the data set has very limited amounts of samples). For this reason, the second part of the evaluation criteria, taken by the Delta Test (DT) is used, as a means of restricting even further the amount of neurons selected for the final model structure.

4. Nonparametric Noise Estimator (NNE): Delta Test

Delta Test (DT) is a non-parametric technique based on nearest neighbors principle. It is a fast scalable algorithm for estimating the noise variance presented in a data set modulo the best smooth model for the data, regardless of the fact that this model is unknown. A useful overview and general introduction to the method and its various applications is given in [27]. The evaluation of the NNE is done using the DT estimation introduced by Stefansson[28].

In the standard DT analysis, we consider vector-input/scalar-output data sets of the form

$$(x_i, y_i | 1 \leq i \leq N) \tag{6}$$

with N the number of samples (observations), and where the input vector $x_i \in \mathbb{R}^d$ is confined to some closed bounded set $C \subset R^d$. The relationship between input and output is expressed by $y_i = f(x_i) + r_i$, where f is the unknown function and r is the noise. The Delta Test estimates the variance of the noise r.

The Delta test works by exploiting the hypothesized continuity of the unknown function f. If two points x and x' are close together in input space, the continuity of f implies that the points $f(x)$ and $f(x')$ will be close together in

output space. Alternatively, if the corresponding output values y and y' are not close together in output space, this can only be due to the influence of noise on $f(x)$ and $f(x)$.

Let us denote the first nearest neighbor of the point x_i in the set $\{x_1, \ldots, x_N\}$ by $x_{f(i)}$. Then the delta test, δ is defined as:

$$\delta = \frac{1}{2N} \sum_{i=1}^{N} \left| y_{f(i)} - y_i \right|^2 \tag{7}$$

where $y_{f(i)}$ is the output of $x_{f(i)}$. For the proof of the convergence of the Delta Test, see [27].

In a word, the Delta test is useful for evaluating relationship between two random variables, namely, input and output pairs. The DT has been introduced for model selection but also for variable (feature) selection: the set of inputs that minimizes the DT is the one that is selected. Indeed, according to the DT, the selected set of variables (features) is the one that represents the relationship between variables and output in the most deterministic way.

In this paper, Delta test is used between the output of the hidden layer and the real output, following the BIC criterion, to further validate the selection of the ELM neurons.

5. DT-ELM

In this section, the general methodology is presented as well as the details of the implementation steps.

5.1. Algorithm

Fig 1 illustrates the main procedures of DT-ELM, and how they interact. In general, DT-ELM is a robust method with one parameter to be tuned. Unlike most of other ELM related methods, DT-ELM has the ability to run without setting expected training error or the maximum number of neurons beforehand, and will automatically stop once the criteria are met. The algorithm of DT-ELM can be summarized as follow:

Given a training set $(x_i, y_i)|x_i \in R^d, y_i \in R, i = 1, 2, ..., d$, activation function $f(x)$. H represents the output matrix of the hidden layer. Each trial cluster contains n neurons.

Initialization step: Let the number of hidden neurons to be zero at the very beginning, then the neurons could be chosen progressively later on by DT-ELM. Set the initial BIC and DT value to be infinite, so that the following steps are always trying to add neurons to DT-ELM to minimize BIC and DT results.

Learning step:

- Randomly generate a cluster of n neurons. n is optional that can be configured according to the different computer power or different data sets. It saves computational time to test neurons cluster by cluster, than one by one. The reason is that along with the selection of the final neurons,

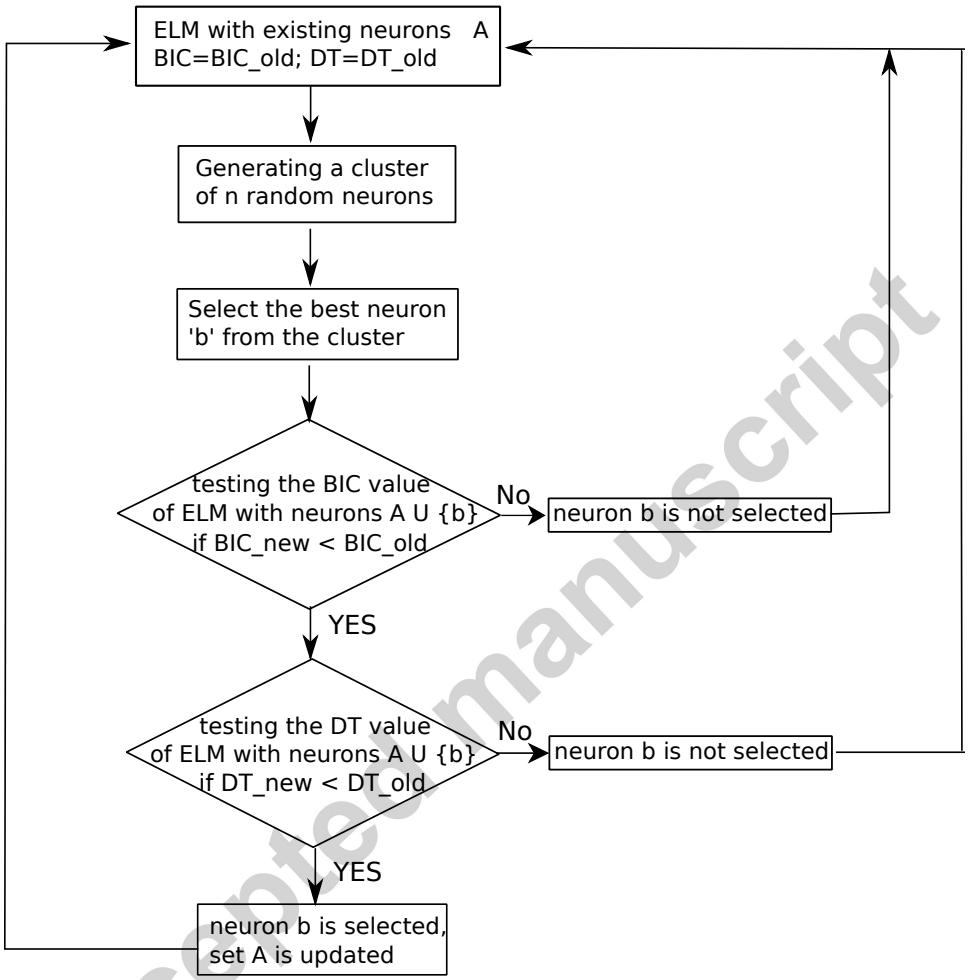

Figure 1: The framework of the proposed DT-ELM

it's more and more difficult to find a neuron that can reduce both BIC and DT value. So it would waste more time to build the ELM one by one.

- Construct ELM using the combination of each of the n neurons and the existing selected neuron set A.(For the first round, it means to construct ELM with each neuron separately as A is null at the begining). Test the BIC value for each new ELM, find the neuron that gives the smallest BIC.
- Check whether the smallest BIC is smaller than the previous BIC value.

If so, continue to next step; otherwise, stop current trial and repeat the learning step. In practice, the value of BIC decreases easily and fast at the beginning, but becomes more and more difficult with the increasing number of neurons.

- Calculate the DT value between the hidden layer and the output for the ELM with the existing neuron and the neuron found in previous step. If the DT results get decreased, this new neuron is added; otherwise stop the current round and repeat the learning step. It is similar with BIC value at the beginning, DT decreased quite fast, but with the increase number of neurons, it becomes extremely difficult to find a new satisfying neuron.

Stop criterion

One advantage of DT-ELM is that only one parameter (number of clusters) needs to be set beforehand, the number of neurons is chosen automatically according to the algorithm. Therefore, when to stop finding new neurons becomes an issue for this method. In this paper, the default setting is 200 extra clusters. The setting of this number is not sensitive to the final performance. As we mentioned that the neurons are tested cluster by cluster, instead of one by one in other incremental learning algorithm. Therefore, this means DT-ELM stop training if DT values doesn't decrease for continuous 4000 new neurons (here each cluster contains $n = 20$ neurons).

5.2. Example

Take data set Bank (more details in Experiment Section) for example, Bank has 8 attributions (variables) and 4499 samples, from which 2999 samples are randomly selected for training and the rest 1500 for test.

Fig 2 illustrates the results of training and testing on bank data using DT-ELM. For this trial, 369 clusters are generate and tested for selection, and 23 neurons are selected eventually.

6. Ensemble modeling

No guideline is always correct. No single method is always the best. This has lead to the idea of trying to combine models into an ensemble rather than selecting among them [29]. The idea seems to work well as demonstrated by many practical applications [30, 31, 32, 33].

6.1. Combining different models into ensembles

It is stated [30] that a particular method for creating an ensemble can be better than the best single model. Therefore, how to combine the models becomes an issue. There are several ways to achieve this. One example is using Non-Negative constrained Least-Squares (NNLS) algorithm [32, 34]. In this paper, we use the equalized weights for all the ensemble models and it works well as shown in the Experiments.

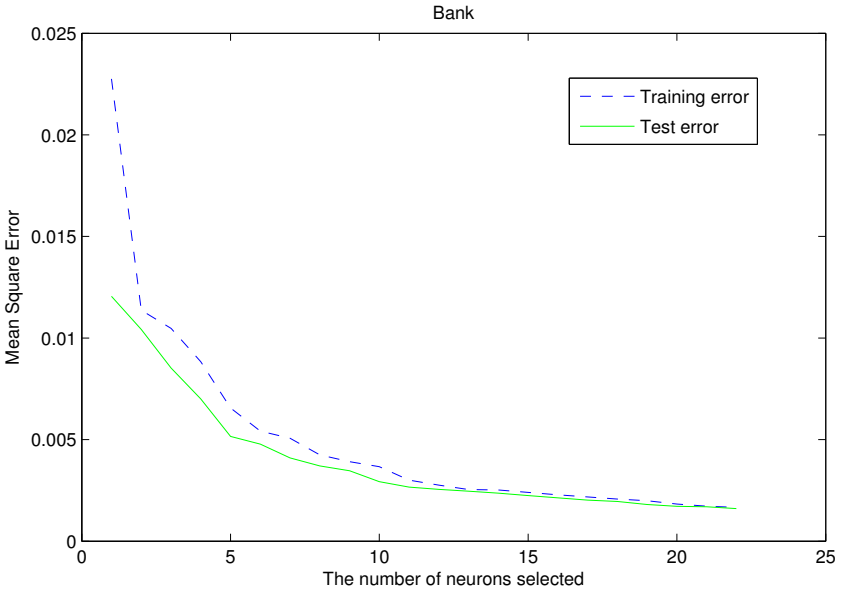

Figure 2: Mean Square Error for Bank, versus the number of Neurons

The ensemble error can be calculated between $y_{Ensemble}$ and y, where $y_{Ensemble} = \sum_{i=1}^{k} \omega_i \hat{y}_i$ is the weighted sum of the output of each i individual models, ω_i is the weighted assigned to the output of the ith model; these weights satisfy $\sum_i \omega_i = 1$; y is the real output of the data and $\hat{y}_i$ is ensemble target. In this paper, we want to build k models and more particularly, $\omega_i = \frac{1}{k}$. Thus, the final output we obtain is $y_{Ensemble} = \sum_{i=1}^{k} \frac{1}{k} \hat{y}_i$.

Fig 3 shows more details on how the ensemble DT-ELM works in this paper. The output of DT-ELM varies, even when using the same training samples, as the hidden layer weights and biases are generated randomly. Therefore, for each training set, 50 models (DT-ELM) are constructed. Then the ensemble step assigns the same weights $\omega = \frac{1}{50}$ to each output of the model y_i. So the training result of the Ensemble DT-ELM is $y_{train} = \frac{1}{50} \sum_{i=1}^{50} y_i$.

6.2. Estimating the performance of the ensemble

Typically, the way to estimate the performance of a method is to divide the data set into training, validation and testing sets. The model is built in the training phase based on the information that the training set contains. The results are validated and the best model is chosen using validation set. Finally, the model is tested in a test set that is not used for building the model.

A validation dataset is not required as DT-ELM can determine the required hidden neurons. Therefore, in this paper, the data set is only divided into

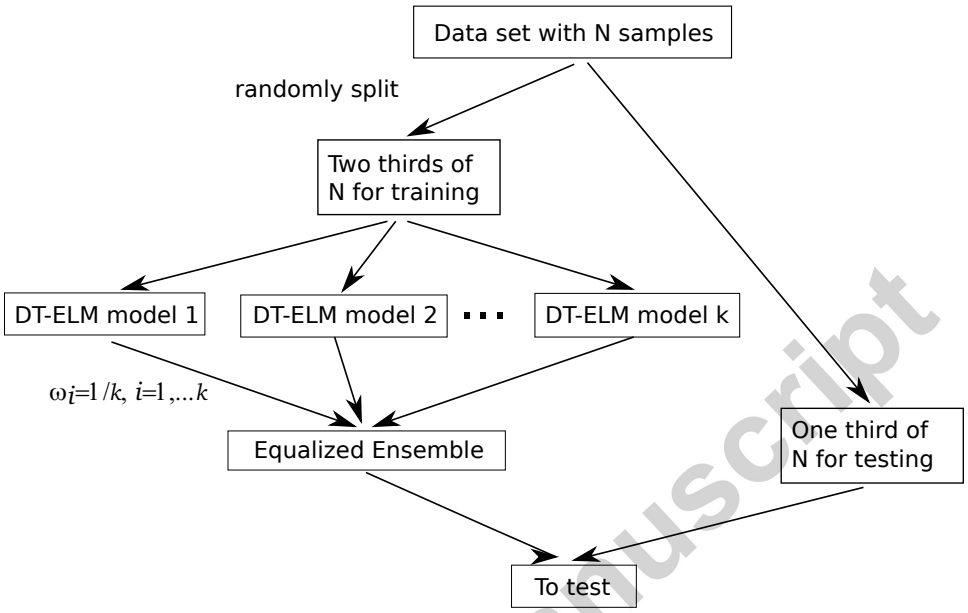

Figure 3: The framework of the proposed Ensemble DT-ELM

training and test set. More particularly, the data set is split randomly ten times with the same proportion to train and test Ensemble DT-ELM, and the final MSE is the average of the 10 repetitions. In this way, we are able to obtain more general performance of the method, the computational time and the standard deviation is reported in next Section.

7. Experiments

In order to compare the proposed Ensemble DT-ELM with other machine learning algorithms, six data sets from UCI machine learning repository [35] are used. They are chosen for their heterogeneity in terms of problem, number of variables, and sizes. Table 1 shows the specification of the 6 selected data sets.

Datasets	# Attributes	# Training data	# Testing data
Ailerons	5	4752	2377
Elevators	6	6344	3173
Servo	4	111	56
Bank	8	2999	1500
Stocks	9	633	317
Boston Housing	13	337	169

Table 1: Specification of the 6 tested regression data sets

The data sets have all been processed in the same way: for each data set, 10 different random permutations are taken without replacement; for each permutation, two thirds are taken for the training set, and the remaining third for the test set (see Table 1). Training sets are then normalized (zero-mean and unit variance) and test sets are also normalized using the same normalization factors. The results presented in the following are hence the average of the 10 repetitions for each data set.

The regression performance of the Ensemble DT-ELM is compared with OP-ELM, ELM and other well-known machine learning methods like Support Vector Machine (SVM), Multilayer Perception (MLP), and Gaussian process for Machine Learning (GPML). The SVM toolbox from [36] is used for the experiments, while the MLP [37] is using the neural network toolbox, part of the Matlab software from the MathWorks, Inc; the GPML toolbox for Matlab from Rasmussen and Williams [38] is used for the GP; and finally, the OP-ELM is performed with the online version of the OP-ELM toolbox and was using a maximum number of 100 neurons.

First are reported in Table 2 the mean square error for the seven algorithms tested. Table 3 and Table 4 illustrate the computational time and the number of neurons selected, respectively. As seen from these tables, the test results of Ensemble DT-ELM are at least as good as that of OP-ELM, with similar computational time, but much simpler model structure (number of neurons) eventually. The number of neurons E.DT-ELM selected is about half of the number with OP-ELM; for some cases, like Ailerons and Elevators, E.DT-ELM uses around 4 neurons instead of about 70 neurons of OP-ELM.

	Ailerons	Elevators	Servo	Bank	Stocks	Boston
E. DT-ELM	3.2e-8	2.1e-6	5.3e-1	1.4e-3	6.6e-1	1.6e+1
	(1.0e-7)	(5.0e-6)	(1.9)	(3.2e-3)	(1.1)	(49)
DT-ELM	8.3e-8	2.5e-6	6.2e-1	1.7e-3	8.5e-1	1.9e+1
	(6.0e-7)	(7.1e-6)	(1.1)	(7.6e-3)	(1.7)	(47)
OP-ELM	2.8e-7	2.0e-6	8.0e-1	1.1e-3	9.8e-1	1.9e+1
	(1.5e-9)	(5.4e-8)	(3.3e-1)	(1.0e-6)	(9.35)	(2.9)
ELM	3.3e-8	2.2e-6	7.1	6.7e-3	3.4e+1	1.2e+2
	(2.5e-9)	(7.0e-8)	(5.5)	(7.0e-4)	(9.35)	(2.1e+1)
GP	2.7e-8	2.0e-6	4.8e-1	8.7e-4	4.4e-1	1.1e+1
	(1.9e-9)	(5.0e-8)	(3.5e-1)	(5.1e-5)	(5.0e-2)	(3.5)
MLP	2.7e-7	2.6e-6	2.2e-1	9.1e-4	8.8e-1	2.2e+1
	(4.4e-9)	(9.0e-8)	(8.1e-2)	(4.2e-5)	(2.1e-1)	(8.8)
SVM	1.3e-7	6.2e-6	6.9e-1	2.7e-2	5.1e-1	3.4e+1
	(2.6e-8)	(6.8e-7)	(3.3e-1)	(8.0e-4)	(9.0e-2)	(3.1e+1)

Table 2: Mean Square Error results for comparison. Standard deviations in brackets

8. Conclusions

This paper proposed a method called Ensemble DT-ELM. It ensembles from a number of DT-ELM models trained with the same training set. BIC and DT is applied into the algorithm with the penalty of the number of the neuron and the estimated variance of the noise between hidden layer and the output. So that

	Ailerons	Elevators	Servo	Bank	Stocks	Boston
E. DT-ELM	5.9	8.7	6.4e-1	3.1e+1	2.1e+1	1.0e+1
DT-ELM	3.2	5.1	2.0e-1	8.3	3.7	2.2
OP-ELM	16.8	29.8	2.1e-1	8.03	1.54	7.0e-1
ELM	9.0e-1	1.6	3.9e-2	4.7e-1	1.1e-1	7.4e-2
GP	2.9e+3	6.5e+3	2.2	1.7e+3	4.1e+1	8.5
MLP	3.5e+3	3.5e+3	5.2e+2	2.7e+3	1.2e+3	8.2e+2
SVM	4.2e+2	5.8e+2	1.3e+2	1.6e+3	2.3e+3	8.5e+2

Table 3: Computational times (in seconds) for comparison

	Ailerons	Elevators	Servo	Bank	Stocks	Boston
E. DT-ELM	3.3	3.7	9.6	19.4	43	33.8
DT-ELM	4.1	4.3	10.2	21.6	51	38.9
OP-ELM	75	74	36	98	100	66

Table 4: Average (over the ten repetitions) of the number of neurons selected for the final model for OP-ELM, DT-ELM and Ensemble DT-ELM

DT-ELM method adds neurons incrementally and stops when couldn't decrease both BIC and DT values.

The significant advantages of this method are its robustness and the sparsity of the model. There is only one parameter need to be tuned and it constructs much more sparse model. As we know that the less hidden nodes used, the more interpretable of the model. On the other hand, ensemble DT-ELM maintains the fast speed even it stops after 4000 unsuccessful test of neurons. These are also proved by the experiments. In the experiments section, six real regression data sets have been tested. The results show that DT-ELM maintains the fast computational time, the good performance, and constructs relatively sparse models.(The number of hidden nodes selected is much less than OP-ELM).

[1] G. B. Huang, Q. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[2] W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, vol. 74, no. 16, pp. 2541–2551, 2011.

[3] N. Liu, Z. Lin, Z. Koh, G.-B. Huang, W. Ser, and M. E. H. Ong, "Patient outcome prediction with heart rate variability and vital signs," *Signal Processing Systems*, vol. 64, no. 2, pp. 265–278, 2011.

[4] Y. Lan, Y. C. Soh, and G.-B. Huang, "Extreme learning machine based bacterial protein subcellular localization prediction," *IJCNN*, pp. 1859–1863, 2008.

[5] Y. Lan, Y. C. Soh, and G.-B. Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72, pp. 3391–3395, 2009.

[6] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *International Journal of Machine Leaning and Cybernetics*, vol. 2, no. 2, pp. 2107–2122, 2011.

[7] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multi-class classification," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2011.

[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proceedings of International Joint Conference on Neural Networks (IJCNN2004)*, vol. 2, (Budapest, Hungary), pp. 985–990, 25-29 July, 2004.

[9] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, vol. 2, (Kunming, China), pp. 1029–1036, 6-9 Dec, 2004.

[10] G.-B. Huang and C.-K. Siew, "Extreme learning machine with randomly assigned RBF kernels," *International Journal of Information Technology*, vol. 11, no. 1, pp. 16–24, 2005.

[11] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "Trop-elm: a double-regularized elm using lars and tikhonov regularization," *Neurocomputing*, vol. 74, no. 16, pp. 2413–2421, 2011.

[12] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "Gpu-accelerated and parallelized elm ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430–2437, 2011.

[13] H. J. Rong, Y. S. Ong, A. H. Tan, and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," *Neurocomputing*, vol. 72, pp. 359–366, 2008.

[14] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: Optimally-pruned extreme learning machine," *In IEEE Transactions on Neural Networks*, vol. 21, pp. 158–162, 2010.

[15] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

[16] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, pp. 3460–3468, 2008.

[17] 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.

[18] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, pp. 576–583, 2008.

[19] C. R. Rao and S. K. Mitra, "Generalized inverse of matrices and its applications," *New York: John Wiley & Sons*, p. 240, 1971.

[20] Y. Lan, Y. Soh, and G. B. Huang, "Constructive hidden nodes selection of extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16-18, pp. 3191–3199, 2010.

[21] 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.

[22] L. Yuan, S. Y.Chai, and G. B. Huang, "Random search enhancement of error minimized extreme learning machine," in *European Symposium on Artificial Neural Networks (ESANN) 2010*, (Bruges, Belgium), pp. 327–332, 2010.

[23] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[24] P. Zhang, "On the convergence rate of model selection criteria," *On the convergence rate of model selection criteria*, vol. 22, pp. 2765–2775, 1993.

[25] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Second International Symposium on Information Theory*, (Budapest), pp. 267–281, 1973.

[26] M. B. Priestley, *Spectral analysis and time series*. Academic Press London ; New York, 1981.

[27] A. J. Jones, "New tools in non-linear modeling and prediction," *Computational Management Science*, vol. 1, no. 2, pp. 109–149, 2004.

[28] A. Stefánsson, N. Koncar, and A. J. Jones, "A note on the gamma test," *Neural Computing and Applications*, vol. 5, no. 3, pp. 131–133, 1997.

[29] J. Cao, Z. Lin, G.-B. Huang, and N. Liu, "Voting based extreme learning machine," *Information Sciences*, vol. 185, no. 1, pp. 66–77, 2012.

[30] S. V. Barai and Y. Reich, "Ensemble modelling or selecting the best model: Many could be better than one," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, pp. 377–386, 1999.

[31] Y. Miche, E. Eirola, P. Bas, O. Simula, C. Jutten, A. Lendasse, and M. Verleysen, "Ensemble modeling with a constrained linear system of leave-one-out outputs," in *In proceedings of 18th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, (Belgium), pp. 19–24, 19-24 Apr, 2010.

[32] Q. Yu, A. Lendasse, and E. Séverin, "Ensemble knns for bankruptcy prediction," in *in CEF 09, 15th International Conference: Computing in Economics and Finance*, (Sydney, Austrilia), June 2009.

[33] L. Kainulainen, Y. Miche, E. Eirola, Q. Yu, B. F. E. Séverin, and A. Lendasse, "Ensembles of local linear models for bankruptcy analysis and prediction," *Case Studies in Business, Industry and Government Statistics (CSBIGS)*, vol. 4, 2011.

[34] C. L. Lawson and R. J. Hanson, "Solving least squares problems," *CLassics in Applied Mathematics*, pp. 245–286, 1995.

[35] A. Frank and A. Asuncion, "Uci machine learning repository," in *http://archive.ics.uci.edu/ml*, 2004.

[36] C. C. Chang and C. J. Lin, "Libsvm: a library for support vector machines," in *Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2001.

[37] C. Bishop, "Neural networks for pattern recognition," in *Oxford University Press*, (USA), 1996.

[38] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," in *The MIT Press*, 2006.