2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Fast and Effective Extreme Learning Machine Algorithm Without Tuning

Meng Joo Er, Zhifei Shao and Ning Wang

Abstract— **Artificial Neural Networks (ANN) is a major machine learning technique inspired by biological neural networks. However, the process of its parameter tuning is usually tedious and time consuming, and thus it becomes a major bottleneck for it being efficiently applied and used by nonexperts. In this paper, a novel ANN algorithm, termed as Automatic Regularized Extreme Learning Machine (AR-ELM), based on a Regularized Extreme Learning Machine (RELM) using ridge regression is proposed. It is a true automatic ANN learning algorithm in the sense that it can automatically identify the appropriate essential system parameter according to the input data without the need of user intervention. Since this method is based on a relatively straightforward formula, it can achieve very fast learning speed. The simulation results shows that the proposed AR-ELM algorithm can achieve comparable results to tedious cross-validation tuned RELM. Furthermore, we also systematically investigate one of the biggest concerns of ELM, its randomness nature, caused by randomly generated parameters.**

I. Introduction

Artificial Neural Network (ANN) is a machine learning paradigm that mimics the function of a human brain. The data explosion in the modern world provides great opportunities for algorithms like ANN that can uncover sophisticated nonlinear relationships in various applications. Although it has been successfully applied to numerous applications, its performance subjects to various conditions. Among those, arguably the network topology, i.e., the number of hidden neurons L, plays the most critical role. This is because other conditions, such as different activations functions and learning rules, have smaller impacts on the performance since the available options are limited. On the other hand, the choices of L typically ranges from tens to hundreds, and the performance of ANN is greatly influenced by this choice and the problems of underfitting and overfitting are commonly associated with it [1].

As a result, for decades researchers carried out various methods to systematically determine the structure and other parameters of ANN. Notable approaches include Genetic Algorithm (GA) [2], where the parameters are incorporated into artificial genes and after mutations and crossover procedures, the optimal set of parameters is selected in a similar way as natural evolution [3]; Particle Swarm Optimization (PSO) [4], where the parameters are optimized in a social psychology fashion [5]; Constructive ANN, where the structure of ANN grows and is pruned according to various criteria [6].

However, these systematical approaches are not truly automatic in a sense that users have to define some parameters for algorithms to proceed. For GA, specific instructions on crossover, mutation and fitness function have to be given; for PSO, a measure of quality has to be defined; for constructive ANN, growing and pruning criteria also have to be known. These automatic tuning methods in a way just transfer the determination of one set of parameters to another, and users still have to make tough choices that may have big impacts to the final algorithm performance. Therefore we argue that the criteria of a true automatic ANN learning algorithm should have the following properties:

- No or very small user intervention
- The limited choices made by the user should have limited impact on the final performance of ANN.

For approaches that failed to meet the above criteria, we believe they are not true automatic algorithms.

The importance of true automatic ANNs seems inconspicuous for scenarios where data is rarely altered, like disease identification or housing price prediction. While for applications such as weather forecast, where the data is rapidly and constantly changing and the prediction has to be made over and over again, it is impractical to refine user choices to adapt to data with different properties. Therefore in order to make ANN more suitable for automatic processes and more user friendly, we aim to design an algorithm that is truly automatic and meets the proposed criteria.

Extreme Learning Machine (ELM) is a variant of feed-forward ANN with a single hidden layer. It has attracted a large amount of research attentions since it has been shown to outperform back-propagation (BP) algorithm and Support Vector Machines (SVM), in terms of learning speed, reliability and generalization [7]. In ELM, the input weights and biases of hidden nodes are randomly generated instead of exhaustive tuned, and therefore it can achieve much faster learning speed than other ANN algorithms [8]. However, one of the biggest concerns towards ELM is also the reason of its popularity: randomness nature, which results in fluctuating ELM performances, caused by different initialization of the hidden nodes input parameters.

Meng Joo Er is currently a Chair Professor with Marine Engineering College, Dalian Maritime University (DMU), China and a Full Professor with School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, and Zhifei Shao is with the Department of Control and Instrumentation, School of Electrical and Electronic Engineering, NTU, Singapore (email: emjer, zshao1@ntu.edu.sg). Ning Wang is with Marine Engineering College, DMU, Dalian, China (email: n.wang.dmu.cn@gmail.com). This work is supported by the National Natural Science Foundation of P. R. China (under Grants 51009017 and 51379002), Applied Basic Research Funds from Ministry of Transport of P. R. China (under Grant 2012-329-225-060), China Postdoctoral Science Foundation (under Grant 2012M520629), Program for Liaoning Excellent Talents in University (under Grant LJQ2013055), and Fundamental Research Funds for the Central Universities of P. R. China (under Grants 2009QN025, 2011JC002 and 3132013025).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

This deficiency of ELM is widely recognized in the ELM community and one common approach to tackle this problem is to use network ensemble techniques [9]: Some base approximators are first trained on the whole or partial data using the ELM algorithm, and various ensemble methods are used in the model fusion. The easiest one is the usage of a simple averaging operator, where the final output is the average result of all the base approximators [10], [11]. Others may favor more complicated approaches such as GA [12], [13] and creating variable learning sets using adaboost or cross-validation [14], [15].

However, comparing with the basic ELM, the ensemble ELM appears to be redundant in structure, since it requires much more base ELMs to create the approximator. Furthermore, the computational time in some ensemble ELM algorithms increases dramatically because of the usage of GA and cross-validation methods, which require a great deal of tuning effort. For example, the topology of each base ELMs in the ensemble has to be defined by the user, and potentially more parameters have to be tuned for algorithms such as GA, adaboost, and cross-validation.

Recently, a new trend in ELM emerges to combine ELM with regularization, specifically, the ridge regression [16]. It was introduced to improve the generalization ability of ELM [7], and has been found that the generalization ability of this Regularized ELM (RELM) is less sensitive to the choice of ridge parameter C and the neuron topology L compared to traditional ELM [17]. And for some activation function, sigmoid for instance, it appears that its generalization performance reaches a plateau rather than deteriorating, when the number of neurons exceeds some value [17]. The benefit of doing so is to transform the selection of ELM structure into the selection of ridge parameter, which is easier to define. Generally speaking, only the ridge parameter needs to be specified by the user, which is now tuned manually through a trial and error manner. As mentioned before, doing so is impractical for automated learning algorithms and human errors might also be involved in the tuning procedure.

In this paper, we first investigate the randomness reduction (improved stability) effect of RELM and demonstrate that RELM can produce consistent results with same topology but different initial input parameters. And thus we argue that RELM is a better alternative compared to ELM ensemble methods in terms of producing stable results. Furthermore, a novel algorithm termed as Automatic Regularized Extreme Learning Machine (AR-ELM) that meets the two criteria to be a true automatic ANN algorithm is proposed. It is based on an analytical way of calculating the ridge parameter C instead of cross-validation, and therefore can achieve very fast learning speed. The remaining of this paper is organized as follows: The preliminaries of ELM, ELM ensemble and RELM are given in Section 2, and the fluctuating performances of respective algorithms are compared in Section 3, and an attempt to explain this randomness reduction effect brought by RELM is also given. The AR-ELM algorithm is introduced in Section 4. Performance evaluation benchmark datasets including 11 regression are done in Section 5. Conclusions are drawn in Section 6.

II. Preliminaries

ELM is a novel feedforward neural networks algorithm with a single hidden layer [7]. Its salient feature is that the input weights and hidden biases are randomly chosen instead of exhaustively tuned, and the output weights are analytically determined using *Moore-Penrose generalized pseudoinverse* [17]. ELM aims to reach smallest training error as well as the smallest norm of output weights. Consequently, it has been reported to provide better generalization performance with much faster learning speed and avoid traditional ANN issues such as learning rate, stopping criterion, number of training epochs and local minima [8], [18], [19]. In this section, the preliminaries of ELM and its variant ELM ensemble and RELM are introduced.

A. Original ELM

The structure of the original ELM is shown in Figure 1.

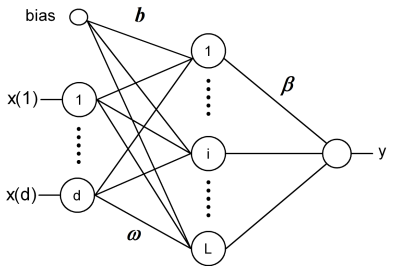

Fig. 1. ELM network structure

The output y with L hidden nodes can be represented by:

$$y = \sum_{i=1}^{L} \beta_i g_i(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\boldsymbol{\omega}_i, b_i, \mathbf{x}) = \mathbf{H}\boldsymbol{\beta} \qquad (1)$$

where $\mathbf{x}, \boldsymbol{\omega}_i \in \mathbb{R}^d$ and g_i denotes the i^{th} hidden node output function $G(\boldsymbol{\omega}_i, b_i, \mathbf{x})$; $\mathbf{H}$ and $\boldsymbol{\beta}$ are the hidden layer output matrix and output weight matrix respectively. In this paper, the Radial Basis Function (RBF) is used as the activation function. For N distinct samples (x_j, t_j), $j = 1, \ldots N$, Eq.(1) can be written as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \qquad (2)$$

where

$$\mathbf{H} = \begin{bmatrix} G(\boldsymbol{\omega}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{\omega}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \vdots & \vdots \\ G(\boldsymbol{\omega}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{\omega}_L, b_L, \boldsymbol{x}_N) \end{bmatrix} \qquad (3)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \end{bmatrix}_{L\times 1} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}_{N\times 1} \qquad (4)$$

where $\mathbf{T}$ is the target matrix.

Since the input weights of its hidden neurons $(\boldsymbol{\omega}_i, b_i)$ can be randomly generated instead of tuned [17], the only

parameters that need to be calculated in ELM is the output weight matrix $\boldsymbol{\beta}$, which can be easily done through Least Squares Estimate (LSE):

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

where $\mathbf{H}^{\dagger}$ is the *Moore-Penrose generalized inverse* of matrix $\mathbf{H}$, which can be calculated through orthogonal projection, where $\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$.

The procedure of ELM goes as follows:

1) Randomly generate hidden neuron parameters $(\boldsymbol{\omega}, \boldsymbol{\beta})$.
2) Calculate hidden layer matrix $\mathbf{H}$ through Eq.(3).
3) Calculate the output weight $\boldsymbol{\beta}$ using Eq.(5).

B. ELM ensemble

The idea of neural network ensembles was first introduced by [9]. By combining the results of an ensemble of neural networks, it has been shown that the overall network performance can be expected to improve. Because of the ease of implementation and relatively low computational requirement, ensemble method has been applied to ELM to reduce its fluctuating performance [10], [14], [20].

The ELM ensemble structure consists of P individual ELMs, where the input parameters $(\boldsymbol{\omega}^j, b^j), j \in [1, P]$ for each one are randomly generated and their output weights $\boldsymbol{\beta}^j$ are analytically determined based on the training data. Although numerous ways exist in generating the final output, the common one is the average of each individual ELM's result [10], for being the easiest and, most of the time, effective approach. Other methods include bootstrap [28] and boosting [21], [22], and even GA [12], [13]. But they may significantly slow down the learning speed of ELM, its most salient feature.

C. Regularized ELM (RELM)

According to ridge regression theory [16], more stable and better generalization performance can be achieved by adding a positive value $\frac{1}{C}$ to the diagonal elements of $\mathbf{H}^T\mathbf{H}$ when calculating the output weight $\boldsymbol{\beta}$ [17], [23]. Therefore, the corresponding RELM becomes:

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C})^{-1}\mathbf{H}^T \tag{6}$$

According to [24], the following matrix inversion property holds:

$$(\mathbf{A} + \mathbf{BCD})^{-1}\mathbf{BC} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1} \tag{7}$$

Let

$$\mathbf{A} = \frac{\mathbf{I}}{C},\ \mathbf{B} = \mathbf{H}^T,\ \mathbf{C} = \mathbf{I},\ \mathbf{D} = \mathbf{H} \tag{8}$$

By substituting Eq.(7,8) into Eq.(6), one can obtain

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C})^{-1}\mathbf{H}^T = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \frac{\mathbf{I}}{C})^{-1} \tag{9}$$

Since the inversion of a matrix with higher dimension requires more computational power, Eq.(6) and Eq.(9) can be selected based on either $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}\mathbf{H}^T$ has smaller dimension.

It has been shown that Eq.(6) and Eq.(9) actually aim at minimizing $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 + \frac{1}{C}\|\boldsymbol{\beta}\|^2$ [17]. Comparing to LSE, where the target is to minimize $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$, an extra penalty term $\frac{1}{C}\|\boldsymbol{\beta}\|^2$ is added to the target of RELM. This is consistent to the theory that smaller output weights $\boldsymbol{\beta}$ play an important role for ELM in achieving better generalization ability [36], [25].

ELM is a very efficient and effective batch learning ANN algorithm and has been implemented in numerous methods and applications. As mentioned before, the salient feature of ELM is its fast learning speed, which is achieved by randomly generating the parameters of its hidden neurons. However, this randomness nature is also one of its biggest problems.

In most ANN learning algorithms, the goal is to minimize the mean square error. We may estimate a model of $\hat{\mathbf{T}}$ of $\mathbf{T}$, with the expected squared prediction error as follows:

$$Err = E[(\hat{\mathbf{T}} - \mathbf{T})^2] \tag{10}$$

The error usually comprises of three elements:

$$\begin{aligned} Err &= [E(\hat{\mathbf{T}}) - \mathbf{T}]^2 + E[\hat{\mathbf{T}} - E(\hat{\mathbf{T}})]^2 + \sigma_e^2 \\ &= Bias^2 + Variance + Irreducible\ Error \end{aligned} \tag{11}$$

Eq.(11) is also known as the bias variance decomposition, where the irreducible error is the noise term in the true relationship and basically cannot be reduced by any model.

ELM and other ANN algorithms that use LSE are unbiased, and do not attempt to reduce the variance term. This is problematic especially for ELM, since the variance is quite high because of its randomness nature. In this case, ridge regression is quite beneficial to ELM, because it forces output weights to smaller values with lower variance, and hence leads to a decrease in prediction error.

The procedure of RELM goes as follows:

1) Randomly generate hidden neuron parameters $(\boldsymbol{\omega}, \boldsymbol{\beta})$.
2) Calculate hidden layer matrix $\mathbf{H}$ through Eq.(3).
3) Calculate the output weight $\boldsymbol{\beta}$ using Eq.(5) with $\mathbf{H}^{\dagger}$ derived from Eq.(6) or Eq.(9).

III. An Investigation of the Inconsistent Performance of ELM, ELM Ensemble and RELM

As mentioned before, the most salient feature of ELM is its extremely fast learning speed, which is primarily obtained by random generalization of input parameters of its hidden nodes. However, this also causes ELM to produce inconsistent results for the same task with different initial parameters, which makes ELM less robust. Arguably, it can be considered as one of the biggest issues exists in the ELM algorithm. In this section, we will probe deeply into the ELM structure and explain what causes the inconsistent performance of ELM. Furthermore, a real world example is given to illustrate the randomness nature of ELM, as well as the performance stabilization effect brought by ELM ensemble and RELM. We will demonstrate that RELM can produce much more consistent results than the ELM and ELM ensemble.

A. A vector view of ELM

From Eq.(3), the $\mathbf{H}$ matrix can be rewritten as:

$$\mathbf{H} = [\mathbf{G}(\mathbf{X}\boldsymbol{\omega}_1 + b_1) \quad \cdots \quad \mathbf{G}(\mathbf{X}\boldsymbol{\omega}_L + b_L)] \tag{12}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_N]^T$. Then the target estimation matrix in Eq.(2) can be rewritten as:

$$\begin{aligned}\mathbf{H}\boldsymbol{\beta} =& [\mathbf{G}(\mathbf{X}\boldsymbol{\omega}_1 + b_1) \quad \cdots \quad \mathbf{G}(\mathbf{X}\boldsymbol{\omega}_L + b_L)]\boldsymbol{\beta} \\ =& \beta_1 G(\mathbf{X}\boldsymbol{\omega}_1 + b_1) + \cdots + \beta_L G(\mathbf{X}\boldsymbol{\omega}_L + b_L) \\ =& \beta_1 \mathbf{G}_1 + \cdots + \beta_L \mathbf{G}_L = \mathbf{T}\end{aligned} \tag{13}$$

The learning process, or the output weights calculation can be considered as finding the best linear combination of $[\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_L]$ to approximate the target vector $\mathbf{T}$. To solve Eq.(13) for any $\mathbf{T}$ with zero error, the linear combination of $\mathbf{G}_i$, $i = 1, 2, \ldots L$, should be able to cover the whole space of $\mathbb{R}^N$ [17]. This means L should be equal or bigger than N, therefore $\mathbf{H}$ can have at least N linearly independent $\mathbf{G}_i$. Given the fact that when $L \geq N$, serious overfitting problem can appear in ELM, L is usually much smaller than N for the ELM optimal structure. Consequently, $\mathbf{H}\boldsymbol{\beta}$ can only approximate $\mathbf{T}$ to a certain degree. With each initialization of ELM, different sets of $\boldsymbol{\omega}$ and $\mathbf{b}$ ($\omega \in [-1, 1]$ and $b \in [0, 1]$) are randomly generated. Even when the same problem is presented, where $\mathbf{X}$ and $\mathbf{T}$ are hold constant, each $\mathbf{G}_i$ will oscillate in the range of $[-G(-\mathbf{X}), G(\mathbf{X}+1)]$ (considering monotonous activation function such as sigmoid is used). Since L is usually set much smaller than N in ELM, the approximation ability to the same $\mathbf{T}$ with different sets of $\mathbf{G}_i$, $i = 1, 2, \ldots L$, ($L < N$) may vary greatly, and this is where the problem of inconsistent performance of ELM stems from.

B. Randomness effects comparison of ELM, ELM ensemble and RELM

To better explain the fluctuating performance of ELM caused by randomly generated hidden layer parameters, Auto MPG dataset is selected for demonstration. Since this comparison is only meaningful for practical structures, the number of hidden neurons L in ELM and the ridge parameter C in RELM are chosen using cross-validation procedure to ensure that optimal models are used (L in RELM is uniformly selected as 1000, same as previous work done by [17]) . Since ELM is a linear-in-the-parameter model, the Leave-One-Out cross-validation procedure can be directly and exactly calculated using the PREdiction Sum of Squares (PRESS) statistics formula [26], [27]:

$$E_{LOO} = \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}_i}{1 - hat_{ii}} \right)^2 \tag{14}$$

where y_i and $\hat{y}_i$ are the i^{th} sample target value and its corresponding estimation, and hat_{ii} is the i^{th} value of the diagonal of the HAT matrix, which transfomrs $\boldsymbol{Y}$ into $\hat{\boldsymbol{Y}}$:

$$\hat{\boldsymbol{Y}} = \mathbf{H}\boldsymbol{\beta} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y} = HAT \cdot \mathbf{Y} \tag{15}$$

Using the LOO formula, the computational time of the cross-validation procedure can be greatly reduced. The specific data configuration is shown in Table. I.

TABLE I
AUTO MPG DATASET

Dataset	Type	L	C	#Attribute	#Train	#Test
Auto MPG	Reg	28	2^{10}	8	261	131

Different from the approaches adopted by most algorithm evaluation methods, where random permutation is used for each run, we keep the training and testing data unchanged in order to solely evaluate the inconsistent performance caused by different initial input parameters. Totally 10 trials are carried out with training and testing data randomly picked from the dataset. Within each trial, 50 runs are done to study the randomness effect with data unchanged, therefore it eliminates the performance fluctuation caused by different data partitions. The ELM ensemble consists of 10 base ELMs and the final output is the average of results from all base ELMs. The simulation results are shown in Figure 2.

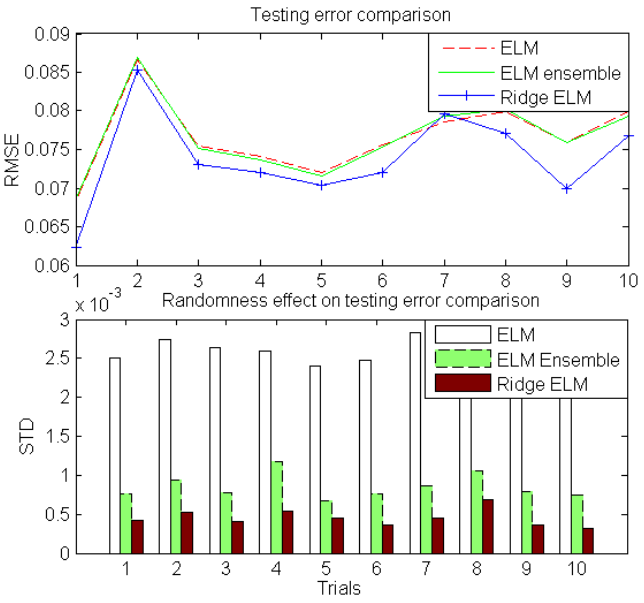

Fig. 2. Randomness effect comparison of original ELM and ELM ensemble with Auto MPG dataset

From Figure 2, it can be seen that although the generalization performance of ELM may not seem to improve by using an ensemble structure , the Standard Deviation (STD) or the randomness, caused by different initial ELM parameters is reduced to about $1/3$ to the original level. Therefore ELM ensemble can indeed achieve more stable performance than using a single ELM. For the case of RELM, It can be seen that not only the testing results are generally improved, the randomness has also been reduced to about half of ELM ensemble and $1/6$ of original ELM. In this Auto MPG problem, the STD is only around half a thousandth of the whole data range, which is a pretty insignificant number. To thoroughly study the randomness

reduction effect of RELM, more comparisons of randomness effect of ELM, ELM ensemble and RELM will be carried out in the performance evaluation section.

C. An Attempt to Explain the Randomness Reduction Effect of RELM

Although the simulation results strongly demonstrate the randomness reduction effect of RELM, it is preferred that this phenomenon can be explained theoretically. We attempt to do this by examining the ELM and RELM from a dimensional perspective.

The output regularization of RELM can suppress the overfitting problem and therefore much more neurons can be used in the network structure than the original ELM. In the previous work done by [17], L is uniformly selected as 1000 for all tasks, which results in $L > N$ for most cases. Intuitively, since more neurons are presented, the linear combination of $\mathbf{G}_i$ should be able to approximate $\left\|\mathbf{H}\boldsymbol{\beta} - \tilde{\mathbf{T}}\right\|$, with $\left\|\mathbf{T} - \tilde{\mathbf{T}}\right\| = \varepsilon$, much easier than ELM, and yields more consistent training performance. And with properly selected ridge parameter C, the inconsistent generalization performance can also be reduced.

To better analyze the randomness reduction effect with varying C and L, the testing performance comparison of ELM and RELM across a wide range of C and L are shown in Figure 3. Since normally $\frac{1}{C}$ takes the value in $[0, 1]$ [29], the C axis in the plots is drawn in a $\log_2$ scale, so that more intensive search can be done in that region.

An interesting phenomenon can be observed in Figure 3a. Generally speaking, by using a small amount of $\frac{1}{C}$, the ELM generalization ability can be improved (same L is used in ELM and RELM), except when L is small (probably before the overfitting effect appears in ELM, and adding C can suppress the overfitting problem). With increasing number of hidden neurons used, RELM can achieve better generalization performance with bigger amount of $\frac{1}{C}$. This demonstrates that RELM is more resistant to the overfitting problem. Although whether optimal RELM performs better than optimal ELM still remains to be determined, the results in Table 3 seem to give an affirmative answer. From Figure 3b, given the same L, RELM has more stable performance, and only a very small region in the plot shows otherwise. In this specific Auto MPG task, RELM already has lower randomness than ELM when $L = 28$, $C = 2^{10}$. Since more neurons tends to offer better approximation ability and smaller randomness effect, and the C is selected so that overfitting effect is limited in RELM, we can almost be certain to say that RELM offers more stable results with carefully chosen parameters, and experiments in the performance evaluation section also show similar results. Therefore we propose the following remark:

Remark 1. Given the same approximation task, optimal L in ELM and C in RELM (L is large enough) are selected, RELM can generally achieve more stable performance. In summary, we believe RELM is a better choice compared

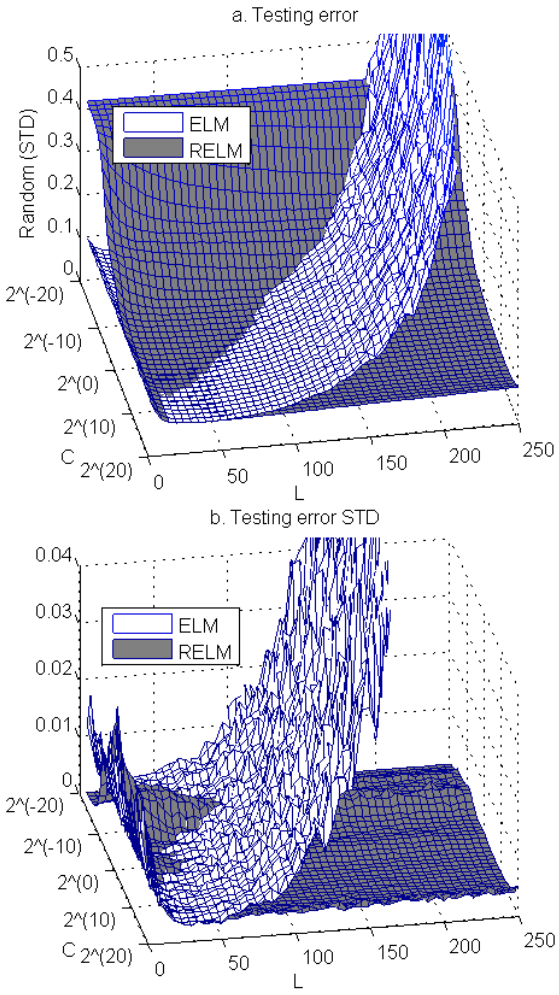

Fig. 3. Testing error and Randomness comparison between ELM and RELM for Auto MPG task

to ELM and ELM ensemble because it offers the following benefits (first two are suggested in previous works done by [17]):

- Better generalization performance may be achieved.
- Only one ridge parameter C needs to be defined by the user.
- The inconsistent performance of ELM caused by random generation of input parameters can be greatly reduced, and even omitted because of its insignificant value.

IV. Automatic Regularized Extreme Learning Machine (AR-ELM)

ANN, as one of the most popular machine learning tool, has been successfully applied to numerous applications because of its non-linear approximation capability. However, to maximize its generalization ability, exhaustive tuning is usually involved. It can be the direct choice of the number of hidden neurons, or the selection of parameters that automatically determine the topology and other parameters of ANN. Whichever way, this process can be time consuming

and it may need certain kind of expertise of the ANN and the specific problem to obtain the optimal set of parameters. Furthermore, it prevents ANN to be widely adopted by fully automatic processes and non-expert users. Therefore it is extremely beneficial to develop an ANN algorithm that meets the criteria of a true automatic algorithm proposed previously. In this section, we propose a novel true automatic ANN algorithm based on ELM and an analytically way of deriving the ridge parameter [30].

Although the cross-validation method provides robust solutions in finding the optimal ridge parameter, it has very high computational requirement, and is not ideal for applications where data is constantly changing and ANN has to be applied iteratively. Even with the convenient LOO formula, the cross-validation process can still be somewhat tedious. Thus it would be very beneficial to preserve the fast learning speed of ELM if the C value can be calculated based on a single RELM structure.

In fact, the research in finding the optimal ridge parameter in ridge regression has already been going on for decades. Besides the computational intensive cross-validation method [31], two main approaches exist, namely graphical driven and data driven methods. As the name implies, the graphical driven methods are based on the plot of some properties of the ridge parameter, including the ridge trace [16], $p-$trace , and Variance Inaction Factor Plot [32]. These graphical driven methods have two main problems. First of all, it requires the users to observe the plot and determine subjectively which C parameter is optimal, thus can be unreliable and time-consuming. Secondly, the plots tend to become messy when the number of hidden nodes is large (one line for each one), which makes them more difficult for users to observe. Furthermore, the computational expensive to generate the plots.

Data driven methods derive the ridge parameter C through a mathematical formula, which analyze the statistical information of the data. The benefit of doing so is quite evident. First of all, they are objective approaches without user intervention, which limit the potential subjective errors and saves human effort. Secondly, they usually can be executed much faster than the cross-validation methods since only one RELM needs to be computed.

Quite a number of algorithms in this category has been proposed, Lawless and Wang formula [30] is chosen AR-ELM because of its superior performance and it is computed as follows:

$$C = \frac{L\hat{\sigma}^2}{\sum_{i=1}^{L} \lambda_i \hat{\alpha}_{LSE,i}^2}, \tag{16}$$

Recall that $\hat{\boldsymbol{\beta}}_{LSE} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}$. Define $\mathbf{W} = \mathbf{XQ}$, where $\mathbf{Q}$ is matrix such that $(\mathbf{HQ})^T(\mathbf{HQ}) = \mathbf{\Lambda} = diag(\lambda_1, \dots \lambda_L)$ and $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. $\lambda_1, \dots \lambda_L$ are the eigenvalues of $\mathbf{H}^T\mathbf{H}$. Then the LSE estimator in the canonical form is calculated as follows:

$$\hat{\boldsymbol{\alpha}}_{LSE} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{T} = \mathbf{\Lambda}^{-1}\mathbf{W}^T\mathbf{T} = \mathbf{Q}^T\hat{\boldsymbol{\beta}}_{LSE} \tag{17}$$

and $\hat{\sigma}^2$ is the estimated variance of regression error, given by:

$$\hat{\sigma}^2 = \frac{(\mathbf{Y} - \mathbf{H}\hat{\beta}_{LSE})^T(\mathbf{Y} - \mathbf{H}\hat{\beta}_{LSE})}{\nu} \tag{18}$$

where $\nu = N - L$, the residual effective degrees of freedom [33].

However, a problem rises when $L > N$, since usually a large value is chosen for L so that the fluctuating performance of ELM can be suppressed, and this can result in the negativity of $\nu = N - L$, which in turn makes Eq.(18) to be negative, an infeasible solution. Cule *et. al* [34] suggest to use the definition of residual effective degrees of freedom [35]:

$$\nu = n - tr(2\tilde{\boldsymbol{H}} - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{H}}^T) \tag{19}$$

where $\tilde{\boldsymbol{H}} = \mathbf{H}\mathbf{H}^{\dagger} = \mathbf{H}(\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C})^{-1}\mathbf{H}^T$. Unfortunately, Eq.(19) cannot be efficiently implemented to find the optimal ridge parameter, since C is also in the formula.

To tackle the negativity problem, we propose to constraint L to be only a proportion of N (e.g., $2/3$) or 1000 (whichever is smaller). L should be selected to be a relatively large value, or else the advantage of regularization effort would be limited ($L = 2N/3$ will make ELM overfit for most of the cases), and the randomness reduction effect brought by a large L can be reduced. The AR-ELM is illustrated in Algorithm 1.

Algorithm 1: Process of AR-ELM

1: Default setting: sigmoid activation function; $L = 2/3N$. If $L > 1000$, then $L = 1000$. Change if required, otherwise continue.
2: Normalize the $\mathbf{H}$ matrix to zero mean and variance one (apply to every hidden node).
3: Decompose $\mathbf{H}_{norm}$: $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = SVD(\mathbf{H}_{norm})$
4: $\mathbf{W} = \mathbf{U} \cdot \mathbf{S}$
5: $\hat{\boldsymbol{\alpha}}_{LSE} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{T}$
6: Find the eigenvalues $\lambda_1, \dots \lambda_L$ of $\mathbf{S}^T\mathbf{S}$
7: $\hat{\sigma}^2 = (\mathbf{T} - \mathbf{W}\hat{\boldsymbol{\alpha}}_{LSE})^T(\mathbf{T} - \mathbf{W}\hat{\boldsymbol{\alpha}}_{LSE})/(N - L)$
8: $C = L\hat{\sigma}^2 / \sum_{i=1}^{L} \lambda_i \hat{\alpha}_{LSE,i}^2$
9: Derive the RELM model using the C and $\mathbf{H}$ calculated above.

As seen from the algorithm illustration, except for the default settings which has limited impact on the final performance, the calculation of the ridge parameter is done through analyzing the statistical information of the training data, without user intervention. Therefore only one RELM model needs to be derived and AR-ELM can run much faster than the cross-validation method.

V. Performance Evaluation

To thoroughly evaluate the performance of AR-ELM, benchmark datasets taken from UCI Machine Learning Repository [37], Statlib [38], and LIBSVM [39] are used, shown in Table II. All simulations are run using Matlab

R2010b on a windows 7 PC with an Intel Core i5 CPU 760 @ 2.80GHz and 8GB RAM. Three variants of the ELM algorithm are selected for comparison, namely original ELM, RELM with C selected using LOO cross-validation, and AR-ELM. Different from other performance evaluation procedure, we also intend to analyze the random effect caused by the different initial parameters of hidden nodes, therefore each test consists of two cycles. In the inner cycle, the training and testing data are randomly selected from the datasets, but hold unchanged for the rest of 30 runs. The L in ELM is chosen using LOO cross-validation, while the C and L in RELM are selected automatically. The outer cycle also consists of 30 runs, but within each one, training and testing data are selected with random permutation. In this way, the error variance of all tests caused by different initial parameters and random permutation of datasets can be both analyzed. Totally 900 runs are done for each dataset. The evaluation results are presented in Table III[1].

TABLE II
SPECIFICATION OF REGRESSION DATASETS

Dataset	Type	L	C	#Attribute	#Train	#Test
Basketball	Reg	5	2^0	5	64	32
Bodyfat	Reg	24	2^0	15	168	84
Auto MPG	Reg	28	2^{10}	8	261	131
Housing	Reg	30	2^5	14	337	169
Forest Fire	Reg	1	2^0	13	345	172
Strike	Reg	19	2^{-5}	7	416	209
Concrete	Reg	139	2^{13}	9	687	343
Balloon	Reg	16	2^{20}	3	1334	667
Quake	Reg	18	2^4	4	1452	726
Space-ga	Reg	69	2^4	7	2071	1036
Abalone	Reg	30	2^0	9	2784	1393

From the results in Table III, it can be seen that AR-ELM offers very competitive results comparing to the original ELM. This is very impressive because it automatically selects the model based on statistical data of RELM and free of user intervention, and there is no need to further divide the training data to evaluate the constructed model. As a result, it also runs much faster than the RELM(F). Although AR-ELM may not be able to offer generalization performance as robust as RELM(F), their results are actually quite close in our tests, which demonstrates that AR-ELM is a reliable approach. Furthermore, this also further strengthen the argument that the number of hidden nodes used by RELM has limited impact on the final performance, as long as it is large enough [2] [17].

It may appear that the training time of ELM is much faster than AR-ELM, and this is because L in the original ELM is usually much smaller than the ones used in AR-ELM. More importantly, the training time does not include the parameter selection procedure. Since there is no upper bound of the optimal L (in the contrary, the bound of C is much easier to define), usually the hidden nodes upper bound is set as 100. Therefore the training time of ELM should at least be multiplied by 100, which results in longer training time than AR-ELM. In the case of manual selection, it could take much longer.

[1]The training time of RELM includes the cross-validation procedure; STD is the error variance in the outer cycle, mainly caused by different training/testing data partition; Random is the error variance in the inner cycle, mainly caused by different initial input parameters. RELM(F) denotes the RELM with full LOO cross-validation procedure.

[2]RELM and AR-ELM employs different L in most cases in our simulation

TABLE III
PERFORMANCE EVALUATION ON REGRESSION DATASETS

Datasets	Algorithm	L	RMSE	STD	Random	Time
Basketball	ELM	7	0.1771	0.0202	0.0175	7.53e-4
	RELM(F)	1000	0.1631	0.0229	6.58e-4	4.1162
	AR-ELM	32	**0.1612**	0.0209	9.00e-4	0.0115
Bodyfat	ELM	15	0.0401	0.0096	0.0085	0.0011
	RELM(F)	1000	**0.0281**	0.0124	3.77e-4	3.2747
	AR-ELM	112	0.0282	0.0127	0.0010	0.0291
Auto MPG	ELM	24	0.0757	0.0051	0.0085	0.0023
	RELM(F)	1000	**0.0736**	0.0064	6.48e-4	4.9914
	AR-ELMs	174	0.0759	0.0058	6.55e-4	0.0718
Housing	ELM	40	0.0754	0.0168	0.0026	0.0071
	RELM(F)	1000	**0.0699**	0.0176	8.51e-4	7.5773
	AR-ELM	225	0.0751	0.0209	2.96e-4	0.1456
Forest Fire	ELM	2	0.0585	0.0249	1.74e-4	7.63e-4
	RELM(F)	1000	**0.0584**	0.0247	2.37e-5	6.2064
	AR-ELM	230	0.0588	0.0238	7.47e-5	0.1250
Strike	ELM	18	0.2981	0.0168	0.0026	0.0071
	RELM(F)	1000	**0.2937**	0.0176	8.51e-4	7.5773
	AR-ELM	278	0.2946	0.0209	2.96e-4	0.1456
Concrete	ELM	83	0.0923	0.0038	0.0038	0.0230
	RELM(F)	1000	**0.0765**	0.0061	0.0017	12.1988
	AR-ELM	458	0.1039	0.0048	8.61e-4	0.6405
Balloon	ELM	20	0.4616	0.8973	0.3420	0.0193
	RELM(F)	1000	**0.0578**	0.0017	1.56e-4	15.0942
	AR-ELM	890	0.1045	0.0035	0.0049	1.8568
Quake	ELM	38	0.1772	0.0090	9.49e-4	0.0490
	RELM(F)	1000	**0.1767**	0.0093	3.34e-5	24.5020
	AR-ELM	968	0.1770	0.0097	4.21e-5	3.9821
Space-ga	ELM	92	0.0360	0.0027	0.0023	0.0686
	RELM(F)	1000	**0.0340**	0.0011	2.49e-4	38.3497
	AR-ELM	1000	0.0356	0.0010	3.79e-5	6.3322
Abalone	ELM	54	0.0792	0.0049	0.0054	0.0259
	RELM(F)	1000	**0.0759**	0.0023	4.91e-4	34.8229
	AR-ELM	1000	0.0762	0.0019	5.67e-5	5.3725

The original ELM also suffers greatly from the fluctuating performance caused by the different initialization of its input parameters. As shown in Table III, the random effect of ELM is usually more than 10 times higher than the rest of the algorithms presented.

VI. CONCLUSION

In this paper, we analyze one of the biggest concerns of ELM algorithm, its fluctuating performance caused by randomly generated parameters of hidden nodes. According to the comparison results, RELM can greatly reduce the randomness effect to a level that can nearly be neglected. Furthermore, RELM can produce even more stable results than an ensemble of 10 base ELMs, which demonstrates that RELM is a better alternative compared to the ELM ensemble in terms of controlling the randomness effect. A

systematic analysis on the reason why RELM can achieve more consistent results than the original ELM is also given. Since basically only the ridge parameter in RELM has a significant influence on its performance, an algorithm named AR-ELM that can automatically select this parameter is proposed.

Instead of further dividing the training data into training and validation sets to evaluate potential model candidates, AR-ELM directly computes the optimal ridge parameter based on the statistical information of the training data, therefore it can achieve much faster speed than the cross-validation methods. Although in theory this approach is not as robust as using the cross-validation procedure, it actually performs quite close in our tests, even better in some cases. This demonstrates that AR-ELM is an effective approach with very fast learning speed.

According to our performance evaluation results, AR-ELM outperform the original ELM (tuned using tedious cross-validation procedure) in almost all tests in terms of generalization performance, and they are much more stable with respect to different initialization parameters. Although the training time of ELM appears to be much faster than the other listed algorithms, the model selection time is not included. By using cross-validation selection procedure, the computational time is usually more than 100 times than the presented results, which is longer than the training time of AR-ELM.

In this paper, we also discuss a very important common problem possessed by almost all ANN algorithms, the need of parameter tuning, which is usually tedious and time consuming. We then suggest two criteria for a true automatic ANN algorithm, and they are met by the proposed algorithm. All the results of the AR-ELM in this paper are produced using the default settings (generally no need to change), meaning nearly zero intervention is needed from users. This makes it much easier to be implemented in the automatic processes and by non-expert users.

REFERENCES

[1] Dreiseitl, S. and Ohno-Machado, L. "Logistic regression and artificial neural network classification models: a methodology review", *Journal of biomedical informatics*, vol 35, pp 353-359, 2002.

[2] Sivanandam, S. and Deepa, S., "Introduction to genetic algorithms", *Springer Publishing Company, Incorporated*, 2007.

[3] Rooij, A., Johnson, R. and Jain, L. "Neural network training using genetic algorithms", *World Scientific Publishing Co., Inc.*, 1990.

[4] Kennedy, J. and Eberhart, R. "Particle swarm optimization", *IEEE proceedings on Neural Networks*, vol 4, pp 1942-1948, 1995.

[5] Zhang, J., Lok, T. and Lyu, M. "A hybrid particle swarm optimization back propagation algorithm for feedforward neural network training", *Applied Mathematics and Computation*, vol 185, pp1026-1037, 2007.

[6] LIU, F. and MENG, J. "A novel efficient learning algorithm for self-generating fuzzy neural network with applications", *International journal of neural systems*, vol 22, pp 21-35, 2012.

[7] Huang, G.-B. and Wang, D. and Lan, Y."Extreme learning machines: a survey", *International Journal of Machine Learning and Cybernetics*, pp 1-16, 2011.

[8] Huang, G.-B., Zhu, Q.-Y. and Siew, C.-K. "Extreme learning machine: Theory and applications", *Neurocomputing*, vol 70, pp 489-501, 1995.

[9] Hansen, L. and Salamon, P. "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 12, pp 993-1001, 1990.

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

[11] Lian, C., Zeng, Z., Yao, W. and Tang, H. "Displacement prediction model of landslide based on ensemble of extreme learning machine" *Neural Information Processing*, pp 240-247, 2012.

[12] Zhu, Q.-Y., Qin, A. K. and Suganthan, P. N. "Evolutionary extreme learning machine", *Pattern Recognition*, vol 38, pp 1759-1763, 2005.

[13] Wang, D. and Alhamdoosh, M., "Evolutionary extreme learning machine ensembles with size control", *Neurocomputing*, 2012.

[14] Liu, N and Wang, H. "Ensemble based extreme learning machine", *Signal Processing Letters, IEEE*, vol 17, pp 527-536, 2010.

[15] Zhai, J.-h., Xu, H.-y. and Wang, X.-z., "Dynamic ensemble extreme learning machine based on sample entropy", *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, pp 1-10, 2012.

[16] Hoerl, A. and Kennard, R. "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, pp 55-67, 1970.

[17] Huang, G.-B., Zhou, H., Ding, X. and Zhang, R. "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, pp1-17, 2010.

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

[19] Huang, G.-B. and Chen, L. "Enhanced random search based incremental extreme learning machine", *Neurocomputing*, vol 71, pp 3460-3468, 2011.

[20] Sun, Z., Choi, T., Au, K. and Yu, Y. "Sales forecasting using extreme learning machine with applications in fashion retailing", *Decision Support Systems*, vol 46, pp 411-419, 2008.

[21] Schapire, R. E. "The strength of weak learnability", *Machine learning*, vol 5, pp 197-227, 1990.

[22] Freund, Y. "Boosting a weak learning algorithm by majority", *Information and computation*, vol 121, pp 256-285, 1995.

[23] Toh, K. "Deterministic neural classification", *Neural computation*, vol 20, pp 1565-1595,

[24] Henderson, H. V. and Searle, S. R. "On deriving the inverse of a sum of matrices", *Siam Review*, vol 23, pp 53-60, 1981.

[25] Bartlett, P. "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol 44, pp.525–536, 1998.

[26] Bontempi, G. and Bersini, H. "Recursive lazy learning for modeling and control," *Machine Learning: ECML-98*, pp. 292-303, 1998.

[27] Myers, R, "Classical and modern regression with applications", *Duxbury Press Belmont, CA*, vol 2, 1990.

[28] Breiman, L. "Bagging predictors," *Machine learning*, vol 24, pp. 123-140, 1996.

[29] Mardikyan, S. and Cetin, E. "Efficient choice of biasing constant for ridge regression", *Int. J. Contemp. Math. Sciences*, vol 3, pp 527-536, 2008.

[30] Lawless, J. and Wang, P. "A simulation study of ridge and other regression estimators", *Communications in Statistics-Theory and Methods*, vol 5, pp 307-323, 1976.

[31] Golub, G. H. and Wahba, G. "Generalized cross-validation as a method for choosing a good ridge parameter", *Technometrics*, vol 21, pp 215-223, 1979.

[32] Draper, N. and Pownell, E. "Applied regression analysis", "Wiley New York", vol 3, 1998.

[33] Halawa, A. and El Bassiouni, M. "Tests of regression coefficients under ridge regression models", *Journal of Statistical Computation and Simulation*, vol 65, pp 341-356, 2000.

[34] Cule, E. and De Iorio, M. "ignificance testing in ridge regression for genetic data", *BMC bioinformatics*, vol 12, pp 372.

[35] Hastie, T. and Tibshirani, R. "Generalized additive models", *Chapman and Hall/CRC*, vol 43, 1990.

[36] Bartlett, P. "For valid generalization, the size of the weights is more important than the size of the network," *Advances in neural information processing systems*, pp. 134–140, 1997.

[37] A. Asuncion and D. N. "UCI machine learning repository", 2007

[38] Vlachos, P. "Statlib project repository", *Carnegie Mellon University*, 2000.

[39] Chang, C.-C. "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol 2, pp 27:1–27:27, 2011.