Robust Input Weight Selection for Well-Conditioned Extreme Learning Machine

Guopeng Zhao[1, 2], Zhiqi Shen[1], and Zhihong Man[3]

[1]Nanyang Technological University, Singapore (639798)
[2]HP Labs Singapore, Singapore (138632)
[3]Swinburne University of Technology, VIC 3122, Australia

{n05008, ascymiao}@ntu.edu.sg, zman@swin.edu.au

Abstract

Recently Extreme Learning Machine (ELM) has been attracting attentions for its simple and fast training algorithm, which randomly selects input weights. Given sufficient hidden neurons, ELM has a comparable performance for a wide range of regression and classification problems. However, in this paper we argue that random input weight selection may lead to an ill-conditioned problem, for which solutions will be numerically unstable. In order to improve the conditioning of ELM, we propose a robust input weight selection algorithm for the ELM with linear hidden neurons. Experiment results show that by applying the proposed algorithm generalization performance is improved in the presence of noises.

Keyword: extreme learning machine, ill-conditioned, least squares.

I. Introduction

Extreme Learning Machine (ELM) [1] which is a single-hidden layer feedforward neural network (SLFNN) randomly selects input weights and hidden neuron biases without training. The output weights are analytically determined by Moore-Penrose generalized inverse. Without iteratively tuning parameters as in Back-Propagation (BP), the learning speed of ELM can be thousands of times faster than gradient-decent learning algorithms. Given enough hidden neurons, performance of this simple learning algorithm is comparable to traditional gradient-decent based algorithms in terms of Root Mean Square Error (RMSE) and classification rate for regression and classification problems respectively.

Many efforts have been emphasized on the accuracy of solutions obtained by ELM, whereas numerical stability is generally ignored [2]. Numerical stability is an important aspect of a system. An ill-conditioned system may have its solutions very sensitive to perturbation in data. In particular, for ill-conditioned system even though the change in the problem data is small, the change in the solution may be large. This implies that the estimated solution from data may be nowhere near the true solution, and thus becomes useless. Unfortunately training of ELM with large hidden neurons usually constitutes an ill-posed problem. Therefore, the solutions obtained by ELM may be sensitive to data perturbation and become a poor estimation to the truth.

In order to improve the conditioning of ELM, this paper proposes an input weight selection algorithm for an ELM with linear hidden neurons. The rest of paper is organized as follows. Section II introduces some important concepts and preliminaries of ELM and condition number which is adopted as a qualitative measure of conditioning. Section III gives the problem formulation and further limits our scope on ELM with linear hidden neurons. Section IV proposes an input weight selection algorithm in light of basic linear algebra. Experiments and numerical results are presented in Section V, and finally this paper concludes with Section VI.

II. Preliminaries

Given a training dataset $L=\{(\boldsymbol{x}(n),\boldsymbol{t}(n)),n=1,...,N\}$, where $\boldsymbol{x}(n)=(x_1(n),...,x_d(n))^T\in\mathbb{R}^d$ and $\boldsymbol{t}(n)=(t_1(n),...,t_m(n))^T\in\mathbb{R}^m$, an ELM with activation function $g(\cdot)$ and $\tilde{N}$ hidden neurons can be analytically modeled as,

$$\sum_{j=1}^{\tilde{N}}\boldsymbol{\beta}_j g(\boldsymbol{w}_j^T\boldsymbol{x}(n)+b_j)=\boldsymbol{t}(n),n=1,...,N \tag{1}$$

where $\boldsymbol{w}_j = (w_{j1},...,w_{jd})^T \in □^d$ is the weight vector connecting the input layer to the jth hidden neuron, b_j is the bias of the jth hidden neuron, and $\boldsymbol{\beta}_j = (\beta_{j1},...,\beta_{jm})^T$ is the weight vector connecting the jth hidden neuron to the output layer. Throughout this paper the output neuron is linear unless otherwise specified. A sketch of an ELM neural network is represented in Figure 1. Note that $z_j(n) = \sum_{i=1}^{d} w_{ji}x_i(n) + b_j$.

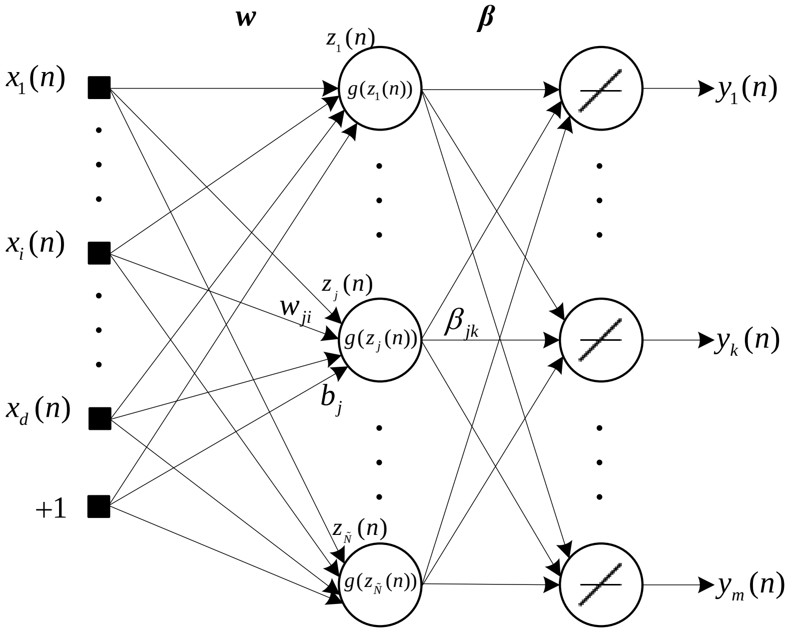


Figure 1. The architecture of an ELM neural network.

Equation (1) can be written in the matrix form as:

$$\mathbf{H\beta} = \mathbf{T} \tag{2}$$

where

$$\mathbf{H} = \begin{bmatrix} g(\boldsymbol{w}_1^T \boldsymbol{x}(1) + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}}^T \boldsymbol{x}(1) + b_{\tilde{N}}) \\ \cdots & \cdots & \cdots \\ g(\boldsymbol{w}_1^T \boldsymbol{x}(N) + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}}^T \boldsymbol{x}(N) + b_{\tilde{N}}) \end{bmatrix} \in □^{N\times\tilde{N}} \tag{3}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}^T \end{bmatrix} \in □^{\tilde{N}\times m} \text{and} \quad \mathbf{T} = \begin{bmatrix} \boldsymbol{t}(1)^T \\ \vdots \\ \boldsymbol{t}(N)^T \end{bmatrix} \in □^{N\times m} \tag{4}$$

Since $\tilde{N} \leq N$ for most cases, the output weights of ELM can be calculated as the least squares solution of linear equation defined in (2), as follows,

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

We argue that the calculation of fining least squares solutions of linear system (5) may constitute an ill-conditioned problem. Specifically, random input weights may lead to an ill-conditioned $\mathbf{H}$, especially when a large number of hidden neurons are used. The conditioning of a matrix can be qualitatively characterized by condition number. Condition number is defined from the analysis of error bounds for linear system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$, and formulated as the product of two matrix norms [3],

$$\kappa(\mathbf{A}) = \| \mathbf{A} \| \cdot \| \mathbf{A}^{-1} \| \tag{6}$$

for any consistent norm. Throughout the paper 2-norm condition number and consistent matrix norms are adapted. In particular for 2-norm we have,

$$\kappa_2(\mathbf{A}) = \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})} \tag{7}$$

By definition $\kappa(\mathbf{A}) = \| \mathbf{A} \| \cdot \| \mathbf{A}^{-1} \| \geq \| \mathbf{A}\mathbf{A}^{-1} \| = \| \mathbf{I} \| = 1$, and $\kappa(\mathbf{A}) = \infty$ for singular matrix. Given $\boldsymbol{x}$ as the least squares solution for $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{x}(\varepsilon)$ for $(\mathbf{A} + \varepsilon\boldsymbol{F})\boldsymbol{x}(\varepsilon) = (\boldsymbol{b} + \varepsilon\boldsymbol{f})$, we would like to see how solution changes according to the perturbations. As shown in [3], for nonsingular square matrix the relative error for $\boldsymbol{x}$ is proportional to $\kappa(\mathbf{A})$,

$$\frac{\|\boldsymbol{x}(\varepsilon) - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \kappa(\mathbf{A}) \left(|\varepsilon| \frac{\|\boldsymbol{F}\|}{\|\mathbf{A}\|} + |\varepsilon| \frac{\|\boldsymbol{f}\|}{\|\boldsymbol{b}\|} \right) + O\left(\varepsilon^2\right) \tag{8}$$

For least squares cases, the error bound is more complicated, and using consistent 2-norm, the relative error is proportional to $\kappa_2(\mathbf{A}) + \| \boldsymbol{r} \|_2 \, \kappa_2(\mathbf{A})^2$ [3], where $\boldsymbol{r} = \mathbf{A}\hat{\boldsymbol{x}} - \boldsymbol{b}$ is the residual for estimated solution $\hat{\boldsymbol{x}}$. An ill-conditioned system has large condition number while a well-conditioned system has small condition number.

For most problems, there is a prior that if ε is small, the change in the solution $\frac{\|\boldsymbol{x}(\varepsilon)-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$ should be small as well. However, it is apparent that for linear system $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$, if $\kappa(\mathbf{A})$ is large the upper bound of relative error will be large, which means relative small perturbation $\mathbf{A}$ in $\boldsymbol{b}$ and may lead to large change in the solution. In practical perturbation due to measurements in the input or approximation during computation is difficult to void. As a consequence, for an ill-conditioned system the calculated least squares $\hat{\boldsymbol{x}}$ is usually a poor estimate to the truth and thus becomes useless. Note that since the error upper bound is affected by the condition number, it does not necessarily mean that the relative error is large for a particular case. Nevertheless, condition number is a good indicator of a system conditioning which shows how close a system is to be ill-conditioned. In this paper we propose an input weight selection algorithm to minimize the condition number of coefficient matrix $\mathbf{H}$, and the problem formulation is shown in the next section.

III. Problem Formulation

One essential learning step of ELM is the least squares calculation for linear system (2). Because from (3) the elements of coefficient matrix $\mathbf{H}$ are functions of input weights $\boldsymbol{w}_i$ and input data $\boldsymbol{x}_j$, the condition number of $\mathbf{H}$ is determined by input weights and input data. Therefore, it is apparent that arbitrary input weights consider no robustness for the solution, and only an elaborate selection of input weights with respect to a particular set of input data may achieve a well-conditioned system. As mentioned before, we adapt condition number of coefficient matrix $\mathbf{H}$ as a qualitative metric of robustness. Combined with ordinary least squares, the problem of network learning is formulated as,

$$\frac{\|\boldsymbol{x}(\varepsilon)-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\le\kappa(\mathbf{A})\left(|\varepsilon|\frac{\|\boldsymbol{F}\|}{\|\mathbf{A}\|}+|\varepsilon|\frac{\|\boldsymbol{f}\|}{\|\boldsymbol{b}\|}\right)+O\left(\varepsilon^2\right) \tag{9}$$

where the first term is the least squares error and the second term indicates the condition number of the coefficient matrix. λ controls the trade-off between residuals of fit and robustness of the solution. When $\lambda\rightarrow 0$, the network finds solutions with more concerns on accuracy, and when

$\lambda \to \infty$ more on robustness instead. The formulated learning can be regarded as a multi-objective optimization problem. For simplicity this paper applies a greedy approach which consists of two consecutive steps of optimization. In light of the layered network structure, the first step optimizes condition number of coefficient matrix $\kappa(\mathbf{H})$, by selecting input weights according to input data $\boldsymbol{x}_j$, and then second step calculates least squares solution $\boldsymbol{\beta}$ as original network does.

However, minimizing the condition number is not trivial since currently we lack of a closed-form expression of $\kappa(\mathbf{H})$ on variable $\boldsymbol{w}$. In this paper, we focus on a simplified SLFNN with only linear hidden neurons. Note that for this case we have $g(z_j(n)) = z_j(n), \forall j$ as in Figure 1. Therefore, we have

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x}(1) + b_1 & \cdots & \boldsymbol{w}_{\tilde{N}}^T \boldsymbol{x}(1) + b_{\tilde{N}} \\ \cdots & \cdots & \cdots \\ \boldsymbol{w}_1^T \boldsymbol{x}(N) + b_1 & \cdots & \boldsymbol{w}_{\tilde{N}}^T \boldsymbol{x}(N) + b_{\tilde{N}} \end{bmatrix} \in \mathbb{R}^{N \times \tilde{N}} \tag{10}$$

Although neural networks with linear activation function are only able to express linear input-output mapping, by limiting our scope on linear neural networks, the study can be analyzed in mathematical detail with linear algebra directly applicable.

IV. Proposed Well-Conditioned Extreme Learning Machine

A. *Robust Input Weight Selection*

Given a matrix $\mathbf{H} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_{\tilde{N}}] \in \mathbb{R}^{N \times \tilde{N}}$ where $\boldsymbol{q}_i$ is the *i*th column vector of $\mathbf{H}$, we can define a correlation matrix such that $\mathbf{A} = \mathbf{H}^{\mathbf{T}}\mathbf{H} \in \mathbb{R}^{\tilde{N} \times \tilde{N}}$. By definition we have $\sigma_i(\mathbf{H}) = \sqrt{\lambda_i(\mathbf{H}^T\mathbf{H})} = \sqrt{\lambda_i(\mathbf{A})}$ where $\sigma_i(\mathbf{H})$ denotes the *i*th singular value of $\mathbf{H}$, and $\lambda_i(\mathbf{A})$ denotes the *i*th eigenvalue of $\mathbf{A}$. Therefore, according to (7) the spectrum condition number can be rewritten as,

$$\kappa_2(\mathbf{H}) = \sqrt{\frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}} \tag{11}$$

where $\lambda_{\max}(\mathbf{A})$ and $\lambda_{\min}(\mathbf{A})$ are matrix $\mathbf{A}$'s largest and smallest eigenvalues respectively.

Let $\boldsymbol{q}_i \perp \boldsymbol{q}_j$ and $\|\boldsymbol{q}_i\|_2 = \|\boldsymbol{q}_j\|_2 = k \geq 0$, , and then we have $\mathrm{A}_{ij} = \boldsymbol{q}_i^T \boldsymbol{q}_j = \begin{cases} k^2 & if\ i = j \\ 0 & if\ i \neq j \end{cases}$. It can be seen that $\mathbf{A}$ is a diagonal matrix of identical diagonal elements. Therefore, it follows that $\lambda_{\max}(\mathbf{A}) = \lambda_{\min}(\mathbf{A})$. From (11) the 2-norm condition number $\kappa_2(\mathbf{H})$ is one, which is the global minimum.

Remark 1: By constructing a coefficient matrix $\mathbf{H}$ of which columns satisfy $\boldsymbol{q}_i \perp \boldsymbol{q}_j$ and $\|\boldsymbol{q}_i\|_2 = \|\boldsymbol{q}_j\|_2 = k \geq 0, \forall i \neq j$, we can guarantee the perfect conditioning of matrix $\mathbf{H}$. As a consequence, the calculated solution of (5) is robust with respect to changes of $\mathbf{H}$ and $\mathbf{T}$.

If linear hidden neurons are considered, coefficient matrix $\mathbf{H}$ can be expressed in the form of $\mathbf{H} = [\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1, \ldots, \tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_{\tilde{N}}] \in \mathbb{R}^{N \times \tilde{N}}$, where $\tilde{\boldsymbol{w}}_i = [\boldsymbol{w}_i^T \ \ b_i]^T \in \mathbb{R}^{(d+1)}$ and $\tilde{\mathbf{X}} = [\tilde{\boldsymbol{x}}(1), \ldots, \tilde{\boldsymbol{x}}(N)] \in \mathbb{R}^{(d+1) \times N}$, by defining $\tilde{\boldsymbol{x}}(n) = [\boldsymbol{x}(n)^T \ \ 1]^T \in \mathbb{R}^{(d+1)}$.

Remark 2: According to Remark 1, we want to adjust $\tilde{\boldsymbol{w}}_i$ for $i = 1, \ldots, \tilde{N}$, such that for $\forall i \neq j$, $\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_i$ is orthogonal to $\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_j$, i.e. $(\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_i)^T (\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_j) = 0$ and $\|\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_i\|_2 = \|\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_j\|_2$.

In order to find input weights for all columns, we consider columns one by one. Given a randomly selected $\tilde{\boldsymbol{w}}_1$, we have $\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1$ as the first column of $\mathbf{H}$. For the second column it is necessary to have $(\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_2)^T (\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1) = 0$ according to Remark 2. Note that $(\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_2)^T (\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1) = \tilde{\boldsymbol{w}}_2{}^T \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1$, and since $\tilde{\mathbf{X}}$ and $\tilde{\boldsymbol{w}}_1$ are known, we can find a vector space S such that $\forall \boldsymbol{a} \in S$, $\boldsymbol{a}$ is orthogonal to $\tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1 \in \mathbb{R}^{d+1}$. This vector space S is also known as the null space of matrix $[(\tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1)^T] \in \mathbb{R}^{1 \times (d+1)}$. Therefore, we can simply choose φ, which is the normalized basis of the null space, and then calculate $\tilde{\boldsymbol{w}}_2 = \dfrac{\|\tilde{\mathbf{X}}^T \tilde{\boldsymbol{w}}_1\|}{\|\tilde{\mathbf{X}}^T \varphi\|} \varphi$ to allow the two columns to have identical norms. For the third column, using obtained $\tilde{\boldsymbol{w}}_1$ and $\tilde{\boldsymbol{w}}_2$ we continue to find $\tilde{\boldsymbol{w}}_3$ such that $\tilde{\boldsymbol{w}}_3$ is orthogonal to both $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_1$ and $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_2$. Similarly, this can be solved by finding the null space of matrix $\begin{bmatrix} (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_1)^T \\ (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_2)^T \end{bmatrix} \in \mathbb{R}^{2\times(d+1)}$.

In this manner, iteratively we can analytically calculate $\tilde{\boldsymbol{w}}_i$ for $i = 1,\dots,\tilde{N}$. However, the existence of $\tilde{\boldsymbol{w}}_i$ should be discussed. Assume that we want to calculate $\tilde{\boldsymbol{w}}_{\tilde{N}}$, and from previous iterations we have $\tilde{\boldsymbol{w}}_i$ for $i = 1,\dots,\tilde{N}-1$. Then $\tilde{\boldsymbol{w}}_{\tilde{N}}$ is supposed to be calculated by solving the null space of matrix $\boldsymbol{\Gamma} = \begin{bmatrix} (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_1)^T \\ \vdots \\ (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_{\tilde{N}-1})^T \end{bmatrix} \in \mathbb{R}^{(\tilde{N}-1)\times(d+1)}$. Assume this matrix is not rank deficient, i.e. $\mathrm{rank}(\boldsymbol{\Gamma}) = \min(\tilde{N}-1, d+1)$. For the dimension of null space of $\boldsymbol{\Gamma}$ we have [3],

$$\dim\left(\mathrm{null}(\boldsymbol{\Gamma})\right) + \mathrm{rank}(\boldsymbol{\Gamma}) = d+1 \tag{12}$$

In order to guarantee the existence of non-empty null space, i.e. $\dim\left(\mathrm{null}(\boldsymbol{\Gamma})\right) > 0$, from (12) we have $d+1-\mathrm{rank}(\boldsymbol{\Gamma}) > 0$. Therefore, $d+2 > \tilde{N}$ and the upper bound of applicable hidden neuron is $d+1$.

As a summary, the algorithm of finding input weights up to $d+1$ is shown in Table 1. After calculating the input weights, output weight is calculated as the same as original ELM.

Table 1. Algorithm for input weight selection

1. Uniformly randomly select $\tilde{\boldsymbol{w}}_1 \in \mathbb{R}^{d+1}$.
2. For $i = 2,...,\tilde{N}$, where $\tilde{N} \le d+1$
3. Calculate $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_1,...,\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_{i-1}$ where $\tilde{\mathbf{X}} = \left[\tilde{\boldsymbol{x}}_1,...,\tilde{\boldsymbol{x}}_N\right] \in \mathbb{R}^{(d+1)\times N}$.
4. Find the null space S of matrix $\begin{bmatrix} (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_1)^T \\ \vdots \\ (\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_{i-1})^T \end{bmatrix} \in \mathbb{R}^{(i-1)\times(d+1)}$.
5. Randomly select a vector φ, where φ is the normalized basis of S.
6. Calculate $\tilde{w}_i$ by $\tilde{\boldsymbol{w}}_i = \dfrac{\left\|\tilde{\mathbf{X}}^T\tilde{\boldsymbol{w}}_{i-1}\right\|}{\left\|\tilde{\mathbf{X}}^T\varphi\right\|}\varphi$.
7. Repeat from Step 2.

B. *Complexity Analysis*

Computational complexity has been investigated to analyze efficiency of the proposed algorithm. According to Table 1, Step 3 requires the computation of matrix $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$. Although it takes $(d+1)^2 N$ operations [3], only one time calculation is required by saving the result. In addition, Step 3 takes $\tilde{N}^2(d+1)^2$ operations to obtain $\tilde{N}$ columns. Assume that both output weights and null space are computed using Singular Value Decomposition (SVD), then number of operations for calculating output weights is $14N\tilde{N}+8\tilde{N}^3$ and $4N\tilde{N}+8\tilde{N}^3$ for null space [3]. Let $\tilde{N} = d+1$, then the upper bound of operations is $11\tilde{N}^4 - 2\tilde{N}^3 + 15\tilde{N}^2 N$ which is $O(\tilde{N}^4 + \tilde{N}^2 N)$. Comparing the complexity of original ELM which is $O(\tilde{N}^3 + \tilde{N}^2 N)$, additional complexity is introduced due to the iterations used for input weight selection. However, the number of iteration is bounded by input dimension d, and the additional complexity can be ignored for most practical problems $N \gg d$. Exceptions are problems for biological data which usually incline to have more input dimensions than number of data.

C. *Algorithm for Well-Conditioned Extreme Learning Machine*

The number of hidden neurons is the only one parameter which needs to be selected. In order to achieve full automation, an adaptive method of choosing number of hidden neurons is proposed based on cross validation.

Thanks to the proposed robust input weight selection, the number of hidden neurons is bounded by $d+1$ for a given dataset $\boldsymbol{x}(n)=(x_1(n),...,x_d(n))^T \in \mathbb{R}^d$, where $n=1,...,N$. This upper bound greatly reduces the search space for optimal number of hidden neurons. Therefore it is possible to conduct an exhaustive search in the reduced space. For the given dataset $\boldsymbol{x}(n)$, there are $d+1$ candidate models, $M_{\tilde{N}}$ for $\tilde{N}=1,...,d+1$. Based on cross validation, the model with smallest testing error will be selected.

Given a dataset $L=\{(\boldsymbol{x}(n),\boldsymbol{t}(n)), n=1,...,N\}$, where $\boldsymbol{x}(n)=(x_1(n),...,x_d(n))^T \in \mathbb{R}^d$ and $\boldsymbol{t}(n)=(t_1(n),...,t_m(n))^T \in \mathbb{R}^m$, the proposed algorithm for Well-Conditioned Extreme Learning Machine is summarized as follows,

Initialization: Let $\tilde{N}=1$, and randomly split L into J equal parts $L_1, L_2, ..., L_J$, where J is an integer. For the jth fold of the J-fold cross-validation, define $L^{(-j)}=L-L_j$ as the training data and L_j as the testing data.

Learning steps:

while $\tilde{N} \le d+1$

(a) Calculate the input weight $\tilde{\boldsymbol{w}}_i$ for $i=1,...,\tilde{N}$ according to Table 1.

(b) Calculate the well-conditioned coefficient matrix $\mathbf{H}$ based on (10).

(c) Calculate the output weight $\boldsymbol{\beta}$ based on (5).

(d) Calculate the overall testing error for the mth single response as follows,

$$\sum_{j}\sum_{(x(i),t_m(i))\in L_j}\left(t_m(i)-M_{\tilde{N}m}^{(-j)}\left(\boldsymbol{x}(i)\right)\right)^2 \tag{13}$$

where a scalar $t_m(i)$ denotes the expected output of the *m*th single response, and $M_{\tilde{N}m}^{(-j)}\left(\boldsymbol{x}(i)\right)$ denotes the *m*th single response for model $M_{\tilde{N}}$ which is trained with $L^{(-j)}$ on input $\boldsymbol{x}(i)$.

end while

(e) choose the optimal number of hidden neurons as,

$$\tilde{N}=\arg\min\sum_{j}\sum_{(x(i),t_m(i))\in L_j}\left(t_m(i)-M_{\tilde{N}m}^{(-j)}\left(\boldsymbol{x}(i)\right)\right)^2 \tag{14}$$

V. Numerical Experiments

In this section, numerical experiments have been conducted to compare the proposed algorithm and original ELM. Both accuracy and robustness of the solutions are considered. Before each trail of experiment, we randomly split the entire data into training data and testing data according to the ratio of 1:1. Totally 80 trials of each dataset are repeated to obtain reliable results and the average over all the trails is used for comparison.

We normalize all the input attributes (except expected outputs) into the range of $[0,1]$. Random variables for both original ELM and the proposed algorithm are uniformly randomly distributed on the interval $[0,1]$. Calculations of least squares and null space of a matrix are implemented by Matlab using SVD. Details of experiments and simulation results are presented in the rest of this section.

A. *Experiment Data*

Breiman's linear regression is used for comparisons between original and proposed algorithm. The regression data is synthesized according to Breiman's work [4] which is commonly used by statisticians as a mouse experiment of testing algorithms. The input vector is a random vector $X = [X_1,..., X_{30}]^T$ which follows a multivariate normal distribution with zero means. The covariance matrix $\mathbf{\Sigma}$ was designed as,

$$\mathbf{\Sigma} = R^{|i-j|}, i, j = 1,...,30 \tag{15}$$

R iss set to be 0.7 for this experiment. The response iss generated according to the linear model,

$$Y = \beta_1 X_1 + ... + \beta_{30} X_{30} + \varepsilon \tag{16}$$

where the noise $\varepsilon \in N(0,1)$ and the coefficients $\beta_m = \gamma \alpha_m$. Given h is a positive integer, $\{\alpha_m\}$ can be defined in three groups,

$$\begin{aligned} &f\ |m-10| < h, \alpha_m = (h - |m-10|)^2; \\ &if\ |m-20| < h, \alpha_m = (h - |m-20|)^2; \\ &if\ |m-30| < h, \alpha_m = (h - |m-30|)^2; \end{aligned} \tag{17}$$

The value of γ is used to control the Signal to Noise Ratio (SNR). Two values 1 and 5 are used for h. For $h = 1$, there are 3 strong nonzero coefficients, and for $h = 5$ there are 23 weak nonzero coefficients. For Breiman's linear regression, 60 observations are generated.

DC motor system identification is simulated in Matlab by constructing a state-space model of DC motor. The model of the DC motor driving an inertial load shows the angular rate of the load as output and applied voltage as input. By varying the applied voltage, we can control the angular rate, and this kind of system is also known as single-input single-output (SISO) linear systems.

We generate a square wave as the input, and feed it into the DC system. The output is simulated and observed. Additional noise is added to the observations. The noise variance is used to control the SNR.

In this experiment the purpose of system identification is to use neural networks to learn the dynamic behavior of the DC motor system from the observed input and system output. For this system, the generation of identification models are well known, and the system equation can be written as, [5]

$$y(k+1)=\sum_{i=0}^{k_1-1}\alpha_i y(k-i)+\sum_{j=0}^{k_2-1}\beta_j u(k-j) \tag{18}$$

where α_i and β_j are constant unknown parameters. The above equation indicates the output of system at time $k+1$ is a linear combination of its past system outputs as well as those of the inputs. Motivated by (18), we rearrange the raw data by allowing each data sample to constitute k_1 past input values and k_2 past output values, labeled with current system output. For DC motor system identification, 400 observations are generated.

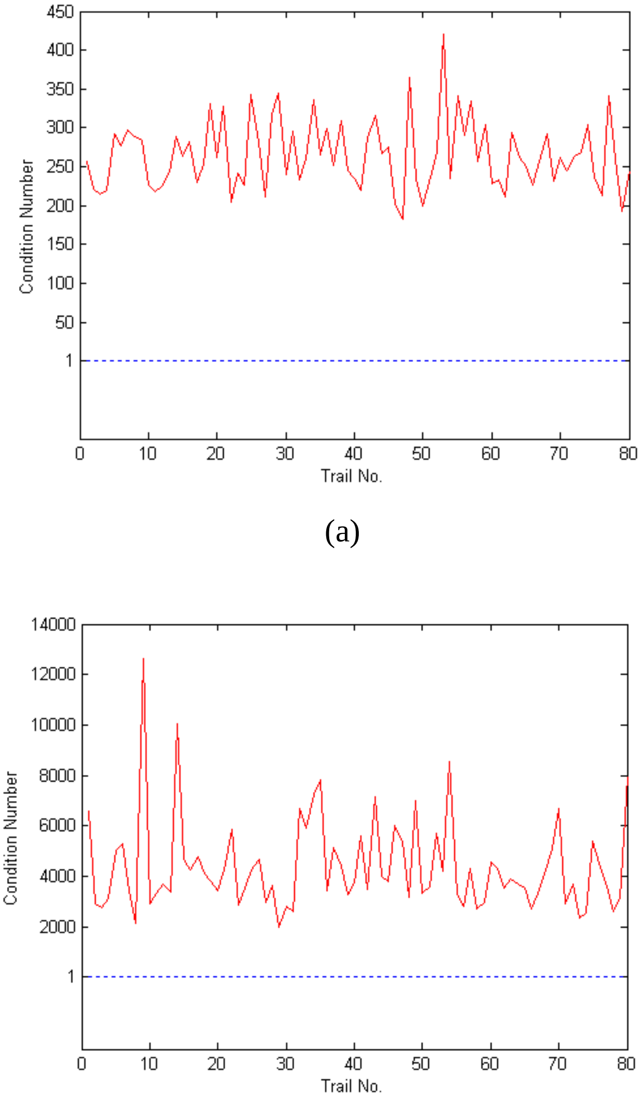
For all experiments, a level of noise is applied to original data to simulate data perturbation, and the noise level is defined by SNR. Throughout the experiments, the SNR for the noise level equals to 0.1 (-10 dB).

B. *Experiment Results*

The performance is evaluated in terms of Root Mean Square Error (RMSE) and size of condition number for accuracy and robustness respectively. Condition number of coefficient matrix $\mathbf{H}$, $\kappa_2(\mathbf{H})$, is calculated using Matlab function *cond()*.

In order to show the effectiveness of proposed input weight selection, the proposed well-conditioned ELM and original ELM use the same number of hidden neurons, unless otherwise specified.

Figure 2 present the comparison of the size of condition numbers between original random and robust input weight selection for ELM networks with 10 and 20 hidden neurons. $h=1$ is set for Breiman's linear regression in this case. For 10 hidden neurons, the proposed algorithm achieves perfect conditioning with $\kappa_2(\mathbf{H})=1$ for all trials, while the ELM with random input weight selection has much larger $\kappa_2(\mathbf{H})$ around 260. If 25 hidden neurons are used for the same data, the figure indicates that ELM with random input weights increases its condition number to 4000, while the proposed ELM maintains its perfect conditioning at $\kappa_2(\mathbf{H}) = 1$ for all trials.



(a)

(b)

Figure 2. Comparison between random (solid line) and proposed (dashed line) input weight selection in terms of the size of condition number, for Breiman's linear regression ($h = 1$), with (a) 10 hidden neurons and (b) 25 hidden neurons

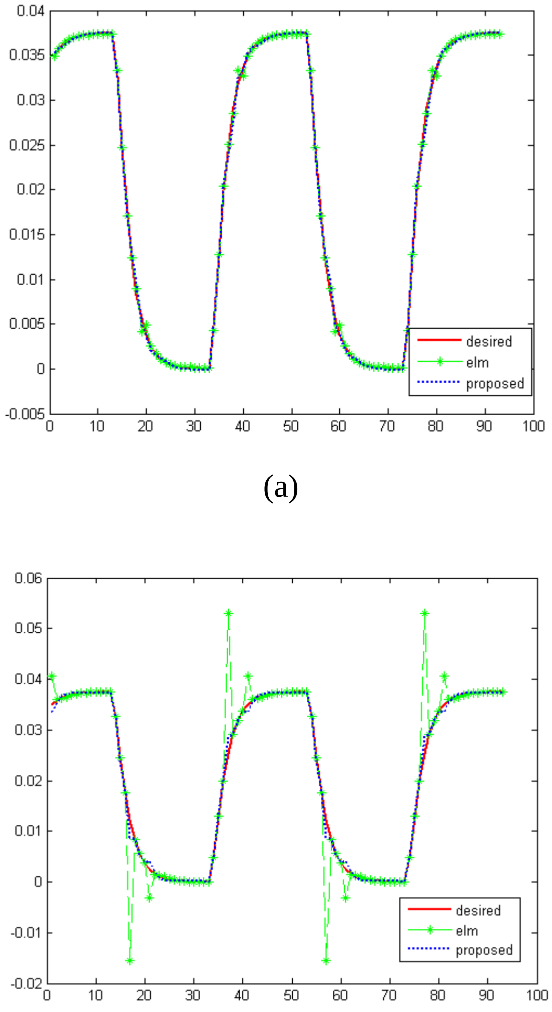
Table 2. Comparison between proposed well-conditioned ELM and original ELM

Data		Algorithm	# Neurons	RMSE or accuracy		SD		Training Time (s)
				Training	*Testing*	*Training*	*Testing*	
Breiman's Linear Regression	$h=1$	ELM	5	0.8862	1.0518	0.0906	0.1272	<1.0e-3
		Proposed		0.8988	1.0201	0.0944	0.1043	0.0016
		ELM	10	0.9096	1.4388	0.1329	0.2011	0.0010
		Proposed		0.8996	1.3596	0.1208	0.1797	0.0035
		ELM	20	0.5673	1.8879	0.1290	0.4887	0.0016
		Proposed		0.5753	1.6827	0.1439	0.3005	0.0113
		ELM	30	0.1935	4.6081	0.1008	9.8393	0.0012
		Proposed		0.1947	3.0146	0.0883	1.2186	0.0259
	$h=5$	ELM	5	1.1916	1.8851	0.1528	0.2853	<1.0e-3
		Proposed		1.1768	1.7529	0.1806	0.2672	0.0018
		ELM	10	1.3096	1.5848	0.1429	0.1896	<1.0e-3
		Proposed		1.3148	1.5273	0.1559	0.1844	0.0058
		ELM	20	0.5472	1.7847	0.0961	0.4408	0.0011
		Proposed		0.5481	1.6073	0.1134	0.3295	0.0138
		ELM	30	0.2077	5.1687	0.1154	5.9266	0.0035
		Proposed		0.2174	3.2329	0.1245	1.4221	0.0226
DC Motor (k_1,k_2)	$(2,3)$	ELM	3	4.731e-3	4.783e-3	1.795e-3	1.873e-3	<1.0e-3
		Proposed		3.567e-3	3.615e-3	1.583e-3	1.530e-3	0.0015
		ELM	5	1.125e-3	1.598e-3	0.578e-3	0.587e-3	<1.0e-3
		Proposed		0.787e-3	1.527e-3	0.452e-3	0.465e-3	0.0018
	$(8,9)$	ELM	10	1.747e-3	1.783e-3	0.844e-3	0.859e-3	0.0012
		Proposed		1.633e-3	1.708e-3	0.648e-3	0.716e-3	0.0025
		ELM	15	0.612e-3	2.656e-3	1.595e-3	1.371e-3	0.0015
		Proposed		0.602e-3	2.463e-3	1.638e-3	1.333e-3	0.0047

The averaged results for 80 trials are presented in Table 2 in different problem settings. As we can see from this table, proposed algorithm achieves similar training results with original ELM algorithm in terms of training RMSE and standard derivation (SD). However, as the improvement in stability, testing performances of well-conditioned ELM outperform those of original ELM in terms of both RMSE and standard derivation. In cases of using larger number of hidden neurons (for example 30 neurons), more than 37% improvement is found in the experiment.

However, the proposed algorithm required more training time than random selection. Table 2 shows that training time for proposed algorithm was around ten times larger than that for original ELM. It is because for this experiment N was not much larger than d . Therefore, additional complexity posed by the proposed algorithm cannot be neglected. Nevertheless, the actual cost of time was still acceptable which was just about a few milliseconds.

Interestingly, for DC motor system identification in Figure 3, ELM can approximate the system step response with small errors without the presence of noise. If a small noise is added to the system, ELM has large glitches in its output. In contrast, proposed algorithm consistently learns the step response with small errors, no matter whether a level of noise is introduced to the system.



(a)

(b)

Figure 3. Comparison between original and proposed algorithms in terms of square response for DC motor system identification ($(k_1 = 2, k_2 = 3)$) (a) without noise and (b) with noise.

VI. Conclusion

This paper first discussed the numerical stability issue of ELM, and argued that random input selection may lead to an ill-conditioned system and thus the solutions may be sensitive to perturbations in the data. In order to improve the conditioning of ELM, this paper proposed an input selection algorithm based on null space calculation. It has been proved that the proposed algorithm can provide perfectly robust solutions up to linear hidden neurons. This paper also compared ELM of random input selection and ELM of proposed input selection though experiment results. It is noticeable that the proposed ELM outperformed original ELM in terms of robustness, and has similar training accuracy in terms of RMSE. It indicates that by applying the proposed algorithm stability is significantly improved. To the contrast, generalization/testing performance is improved especially when noise is presented.

References

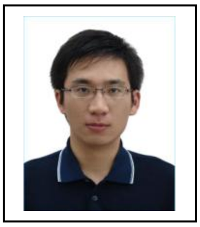
[1] G.-B. Huang, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," in Proc International Joint Conference on Neural Networks, 2004.

[2] G. Zhao, Z. Shen, C. Miao, and R. Gay, "Enhanced Extreme Learning Machine with Stacked Generalization," in Proc. International Joint Conference on Neural Networks, 2008, pp. 1192-1199.

[3] G. H. Golub and C. F. Van Loan, Matrix computations: Johns Hopkins University Press, 1996.

[4] L. Breiman, "Stacked regressions," Machine Learning, vol. 24, pp. 49-64, 1996.

[5] Narendra K. S. and Parthasarathy K., "Identification and control of dynamical systems using neural networks," IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council, vol. 1, p. 4, 1990.



Guopeng Zhao received the B.E. degree (with first-class hons.) in computer engineering from Nanyang Technological University, (NTU), Singapore, in 2006. Since 2006, he has been working towards the Ph.D degree in electrical and electronics engineering at NTU.

His research interests include neural networks, intelligent agents, and Grid/Cloud computing.

Chunyan Miao received the B.S. degree in computer science from Shandong University in 1988 and the M.Eng. and Ph.D. degrees in computer engineering from Nanyang Technological University (NTU), Singapore, in 1996 and 2001, respectively.

She has been an Assistant Professor at the School of Computer Engineering (SCE), Nanyang Technological University (NTU), since 2003. She was an Instructor/Postdoc Fellow at Simon Fraser University, Canada, prior to joining SCE/NTU. Her main research focuses on the study of human factors of agents, social–ecological models of multiagent systems, and their applications in real world systems.

Zhihong Man received the B.E. degree from Shanghai Jiaotong University, Shanghai, China, the M.S. degree from Chinese Academy of Sciences, China, and the Ph.D. degree from The University of Melbourne, Melbourne, Australia, all in electrical and electronic engineering, in 1982, 1996, and 1993, respectively.

From 1994 to 1996, he was Lecturer in the Department of Computer and Communication Engineering, Edith Cowan University, Perth, Australia. From 1996 to 2001, he was Lecturer and then Senior Lecturer in the School of Engineering, The University of Tasmania, Australia. In 2001, he was Visiting Senior Fellow in the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. From 2002 to 2007, he was Associate Professor of Computer Engineering at NTU. Since November 2007, he has been with the School of Engineering, Monash University Sunway Campus, Malaysia, where he is the Professor of Electrical and Computer Systems Engineering, the Chair of the School Research Committee and the Chair of Monash University Sunway Campus Research Committee. His research interests are in neural networks, fuzzy systems, time-varying systems, nonlinear systems, signal processing, robotics, intelligent control, and transmission control protocol (TCP) congestion control. He has published more than 140 journal and conference papers in these areas.