Analysis on fast training speed of extreme learning machine and replacement policy

Shi-Xin Zhao*

College of Management,
Hebei University,
Baoding 071002, China
and
Department of Mathematics and Physics,
Shijiazhuang Tiedao University,
Shijiazhuang 050043, China
Email: cssxzhao@163.com
*Corresponding author

Xi-Zhao Wang

College of Computer Science and Soft Engineering,
Shenzhen University,
Shenzhen 518060, China
Email: xizhaowang@ieee.org

Li-Ying Wang and
Jun-Mei Hu

Department of Mathematics and Physics,
Shijiazhuang Tiedao University,
Shijiazhuang 050043, China
Email: wly_sjz@sohu.com
Email: hu_junmei1979@163.com

Wei-Ping Li

School of Economics and Management,
Beijing Jiaotong University,
Beijing 100044, China
Email: cppsulwp@163.com

Abstract: Extreme learning machine is known for its fast learning speed while maintaining acceptable generalisation. Its learning process can be divided into two parts: (1) randomly assigns input weights and biases in hidden layer, and (2) analytically determines output weights by the use of Moore-Penrose generalised inverse. Through the analysis from theory and experiment aspects we point out that it is the random weights assignment rather than the analytical determination with generalised inverse that leads to its fast training speed. In fact, the calculation of generalised inverse of hidden layer output matrix based on singular value decomposition (SVD) has very low efficiency especially on large scale data, and even directly cannot work. Considering this high calculation complexity reduces the learning speed of ELM conjugate gradient is introduced as a replacement of Moore-Penrose generalised inverse and conjugate gradient based ELM (CG-ELM) is proposed. Numerical simulations show that, in most cases, CG-ELM achieved faster speed than ELM in the condition of maintaining similar generalisation. Even in the case that ELM cannot work because of the huge amount of data CG-ELM attains good performance, which illustrates that Moore-Penrose generalised inverse is not the contribution of fast learning speed of ELM from experiment view.

Keywords: extreme learning machine; generalised inverse; SVD; conjugate gradient method.

Reference to this paper should be made as follows: Zhao, S-X., Wang, X-Z., Wang, L-Y., Hu, J-M. and Li, W-P. (2017) 'Analysis on fast training speed of extreme learning machine and replacement policy', *Int. J. Wireless and Mobile Computing*, Vol. 13, No. 4, pp.314–322.

Biographical notes: Shi-Xin Zhao is a PhD candidate at Hebei University. Her main research interests include neural networks, machine learning and pattern recognition.

Xi-Zhao Wang is PhD supervisor, vice director of BDI, IEEE Fellow, and Editor-in-Chief of IJMLC. His main research interests are machine learning and uncertainty information processing.

Li-Ying Wang is a Professor at Shijiazhuang Tiedao University. Her research interests are reliability Engineer, stochastic models, applications of probability and statistics.

Jun-Mei Hu is a Lecturer at Shijiazhuang Tiedao University. Her main research interests include algebra and the history of modern mathematics.

Wei-Ping Li is a post-doctoral at Beijing Jiaotong University. His research interests include network security, Internet finance.

1 Introduction

Owing to its strong nonlinear mapping ability, robustness, and self-learning ability, feed-forward neural networks have been extensively studied (Kwok and Yeung, 1997; Liang et al., 2006; Zhang and Suganthan, 2016), and have been widely used in various areas of machine learning in the past decades (Qiu et al., 2016; Jamli et al., 2015; Dai and Chen, 2016; Wang and Tian, 2015). Single hidden layer feed-forward neural networks (SLFNs), as one of the most popular feed-forward neural networks, have been extensively studied on their learning capabilities and fault tolerant abilities (Guliyev and Ismailov, 2016; Ding et al., 2015; Cao et al., 2016; Wang et al., 2016). The most widely used learning algorithm for feed-forward neural networks is called back propagation (BP) algorithm (Marquardt, 1963; Werbos, 1994). Although Funahashi and Cybenko had proven that SLFNs trained with BP algorithm have the ability to approximate any continuous function with arbitrary precision, this algorithm is relatively slow since all the parameters of SLFNs need to be tuned through iterative procedures and easy to fall into local minima.

In order to overcome these deficiencies of BP algorithm, extreme learning machine (ELM) was proposed by Huang et al. (2004, 2006). This algorithm is used to train single hidden layer feed-forward neural networks, in which weights linking input layer and hidden layer and biases in the hidden layer are chosen randomly, while weights linking hidden layer and output layer are analytically determined though Moore-Penrose generalised inverse. For its fast learning speed, ELM has become a hot topic in the field of neural networks and has attracted the interest of more and more researchers (Emilio et al., 2011; Mohammed et al., 2011; Huang et al., 2012; Liu et al., 2015; Zhao and Wang, 2014; Wu et al., 2011; Zhu et al., 2005). The most notable feature of ELM is the extremely fast learning speed while maintaining acceptable generalisation. Explaining why ELM is so fast is significant. In this paper we aim to give some precise explanations for its fast learning speed and try to give a replacement policy for this algorithm.

The learning process of ELM can be divided into two parts, the first part is randomly choosing the weights linking input layer and hidden layer and biases in hidden layer, the second part is analytically determining the weights between hidden layer and output layer with Moore-Penrose generalised inverse (Huang et al., 2006). From both theory and experiment aspects, this paper points out that it is the first part rather than the second part that leads to the extremely fast learning speed. Moreover, the computational of Moore-Penrose generalised inverse even limits the learning speed of ELM. The reason is the calculation of generalised matrix depends on singular value decomposition (SVD), whose computational complexity is extremely high so as to seriously restrict training speed of ELM even directly lead the inability on some large scale data.

Among many optimisation algorithms, such as gradient descent (Mason et al., 1999), Newton's method (Grippo et al., 1986), conjugate gradient method (Hestenes and Stiefel, 1952), gravitational search algorithm (Guo et al., 2016), etc., we chose conjugate gradient method as a replacement for Moore-Penrose generalised inverse. That is because conjugate gradient method has low calculation complexity, less storage space and fast convergence speed (Hestenes et al., 1952). In order to overcome the high complexity of output layer generalised inverse matrix in ELM, this paper proposes conjugate gradient based extreme learning machine (CG-ELM). This method uses conjugate gradient method instead of Moore-Penrose generalised inverse to determine the output weights of the network so as to avoid the high complexity computation of generalised inverse. The new proposed learning algorithm shows obvious superiority than basic ELM in speed while maintaining similar generalisation, even gives a good performance on a big data that cannot operate under ELM frame on PC machine. As the fast training speed and convenient incremental implementation, CG-ELM has great potential for the establishment and analysis of large scale data.

This paper is organised as follows. Section 2 gives a brief review of ELM and analyses the complexity of SVD. Section 3 further proposes the CG-ELM learning algorithm after a simple introduction of conjugate gradient method. Performance evaluation of ELM and the new proposed algorithm are given in Section 4. Section 5 gives the discussions and conclusion.

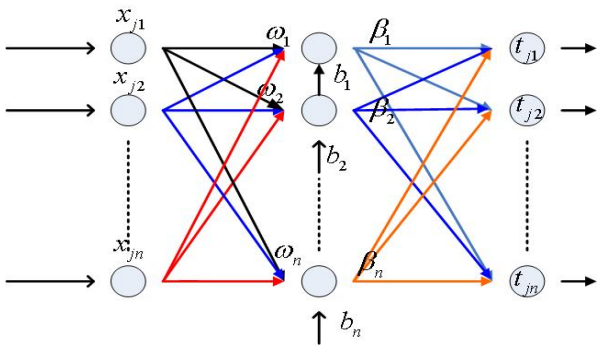
2 Brief review of extreme learning machine

In this section, we will give a brief review of extreme learning machine (ELM) proposed in 2004 and 2006 by Huang et al. (2004, 2006).

2.1 Extreme learning machine

ELM is a fast learning algorithm for SLFNs, the structure is as Figure 1.

Figure 1 Structure of ELM



The mathematical model of standard SLFNs with $\widetilde{N}$ hidden nodes and activation function $g(x)$ is:

$$\sum_{i=1}^{\widetilde{N}}\beta_i g_i(x_j)=\sum_{i=1}^{\widetilde{N}}\beta_i g(\omega_j\cdot x_j+b_i)=t_j \qquad j=1,2,\cdots,N, \quad (1)$$

where (x_j,t_j) are N arbitrary distinct samples in database, $x_j=(x_{j1},x_{j2},\cdots,x_{jn})^T\in R^n$ is input vector of the *j*th sample, $t_j=(t_{j1},t_{j2},\cdots,t_{jm})^T\in R^m$ is output vector of the *j*th sample. $\omega_i=(\omega_{i1},\omega_{i2},\cdots,\omega_{in})$ is the weight vector linking the *i*th hidden node to the input nodes, $\beta_i=(\beta_{i1},\beta_{i2},\cdots,\beta_{im})$ is the weight vector linking the *i*th hidden node to the output nodes, and b_i is the threshold of the *i*th hidden node. $\omega_i\cdot x_j$ denotes the inner product of ω_i and x_j.

The above *N* equations can be compactly written as:

$$H\beta=T, \quad (2)$$

where

$$H(\omega_1,\cdots,\omega_{\widetilde{N}},b_1,\cdots,b_{\widetilde{N}},x_1,\cdots,x_N)=\begin{pmatrix} g(\omega_1\cdot x_1+b_1) & \cdots & g(\omega_{\widetilde{N}}\cdot x_1+b_{\widetilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\omega_1\cdot x_N+b_1) & \cdots & g(\omega_{\widetilde{N}}\cdot x_N+b_{\widetilde{N}}) \end{pmatrix}_{N\times\widetilde{N}}, \quad (3)$$

$$\beta=\begin{pmatrix}\beta_1^T\\ \vdots\\ \beta_{\widetilde{N}}^T\end{pmatrix}_{\widetilde{N}\times m},\ T=\begin{pmatrix}t_1^T\\ \vdots\\ t_N^T\end{pmatrix}_{N\times m}. \quad (4)$$

H is called the hidden layer output matrix of the neural networks, the *i*th column of *H* is the *i*th hidden node output with respect to input $(x_1,\cdots,x_N)$.

Huang had proved that in such networks not all weights need to be adjusted (Huang et al., 2006). In fact, the input weights and biases in hidden layer needn't be adjusted. Once randomly given at the beginning of the study the hidden layer output matrix H will not change. Since H is an over-determined matrix (the sample number is much larger than the number of hidden layer nodes so the number of rows is much larger than the number of columns), what we need to do is to look for a least-squares solution of equation (2), that is to find $\hat{\beta}$ to satisfy:

$$\left\|H(\omega_1,\cdots,\omega_{\widetilde{N}},b_1,\cdots,b_{\widetilde{N}})\hat{\beta}-T\right\| = \min_{\beta}\left\|H(\omega_1,\cdots,\omega_{\widetilde{N}},b_1,\cdots,b_{\widetilde{N}})\beta-T\right\|. \quad (5)$$

This linear system can be solved by Moore-Penrose generalised inverse as follows, the smallest norm least-square solution of this above linear system is:

$$\hat{\beta}=H^{\dagger}T, \quad (6)$$

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

ELM can be summarised as follows (Huang et al., 2006):

Algorithm ELM:

Input: Training set $X=\{(x_j,t_j)\,|\,x_j\in R^n,t_j\in R^m\}$, $j=1,2,\cdots,N$, activation function $g(x)$ and hidden node number $\widetilde{N}$.

Output: Output weights of SLFNs.

Step 1: Randomly assign input weights ω_i and biases b_i, $i=1,2,\cdots,\widetilde{N}$;

Step 2: Calculate the hidden layer output matrix H;

Step 3: Calculate the output weights as $\hat{\beta}=H^{\dagger}T$.

From description above we know the process of ELM can be divided into two parts, the first is randomly assigning input weights and biases in the hidden layer, the second is computing output weights by using Moore-Penrose generalised inverse. The reason why Moore-Penrose generalised inverse is selected may be that solution obtained by Moore-Penrose generalised inverse is not only the least-squares solution but also the smallest norm one. It had been proved that smaller the norm of weights is, better generalisation performance of the networks tend to have (Bartlett, 1998). Although Moore-Penrose generalised inverse gives the smallest norm least-squares solution, it has high computational complexity at the same time, especially on high order matrix. Sometimes it even means ELM cannot operate on some large scale data as demonstrated in Section 4.

2.2 Computational complexity analysis of SVD

In ELM, the Moore-Penrose generalised inverse is obtained by SVD, we will analyse the computational complexity as following.

- *Time complexity*: The calculation of SVD is a hard problem, whose time complexity is $O(n^3)$, and will be

especially slow for dense mass matrix. With the growth of matrix size, it becomes difficult to solve eigenvalue, so the computational complexity of SVD is the third power of growth;

- *Space complexity*: The calculation of SVD needs a large amount of storage space, especially for large scale matrix, the storage space requirement in ELM is not acceptable.

Above analysis illustrates the high computational complexity of SVD from theory aspect, which will be described on some large scale data from experimental aspect in Section 4. We try to find a new method to verify our judgment with the way of replacing SVD by conjugate gradient algorithm in the output weights computation in ELM.

3 Conjugate gradient based ELM

In this section, we will first briefly introduce the conjugate gradient method, and then propose our new method.

3.1 Conjugate gradient method

Conjugate gradient method is an iterative method for positive definite coefficient matrix of linear equation

$$Ax = b . \tag{7}$$

To resolve this problem is to find the minimum solution of quadratic function

$$\varphi(x) = \frac{1}{2}x^T Ax - b^T x . \tag{8}$$

The basic idea of conjugate gradient method is to conjugate the negative gradient direction of current point and the last searching direction, and set it as the next searching direction of current point. Conjugate gradient algorithm can be described as follows (Hestenes and Stiefel, 1952):

Algorithm conjugate gradient:

Step 1: Given coefficient matrix A, target vector b and precision standard ε. Set initial value and iteration count as:

$$x^{(0)} = 0,\ r^{(0)} = b - Ax^{(0)},\ d^{(0)} = r^{(0)},\ k = 0; \tag{9}$$

Step 2: Calculate

$$\alpha_k = \frac{\left(r^{(k)}\right)^T r^{(k)}}{\left(d^{(k)}\right)^T Ad^{(k)}}, \tag{10}$$

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}, \tag{11}$$

$$r^{(k+1)} = r^{(k)} - \alpha_k Ad^{(k)}; \tag{12}$$

Step 3: If satisfied $\left\|x^{(k+1)} - x^{(k)}\right\| < \varepsilon$, or $k = n-1$, stops, and $x^{(k+1)}$ is the solution of equation $Ax = b$, otherwise calculate

$$\beta_{k+1} = \frac{\left(r^{(k+1)}\right)^T r^{(k+1)}}{\left(r^{(k)}\right)^T r^{(k)}}, \tag{13}$$

$$d^{(k+1)} = r^{(k+1)} + \beta_{k+1} d^{(k)}; \tag{14}$$

Step 4: Set $k = k+1$, and go back to step 2.

Conjugate gradient algorithm serves for positive definite coefficient matrix, but the linear system (2) in ELM is an over-determined matrix, so we change to find the key of normal equation of (2):

$$H^T H\beta = H^T T . \tag{15}$$

The relation of original equation and normal equation is described as follows:

Theorem 3.1: *Let x^* be the least-squares solution of over-determined equation $Ax = b$, then it is necessary and sufficient that x^* is the solution of equation $A^T Ax = A^T b$.*

Proof: Sufficiency: Let there exist *n*-dimension vector x^* satisfied $A^T Ax^* = A^T b$, take a *n*-dimension vector $x \neq x^*$ arbitrarily, let $y = x - x^*$, then $y \neq 0$, and

$$\begin{aligned}
&\|b - Ax\|_2^2 = \|b - Ax^* - Ay\|_2^2 \\
&= (b - Ax^* - Ay, b - Ax^* - Ay) \\
&= (b - Ax^*, b - Ax^*) - 2(Ay, b - Ax^*) + (Ay, Ay) \\
&= \|b - Ax^*\|_2^2 - 2y^T A^T (b - Ax^*) + \|Ay\|_2^2 \\
&= \|b - Ax^*\|_2^2 + \|Ay\|_2^2 \geq \|b - Ax^*\|_2^2 \quad .
\end{aligned}$$

So x^* is the least-squares solution of equation $Ax = b$.

Necessity: The *i*th component of vector $r = b - Ax$ is:

$$r_i = b_i - \sum_{k=1}^{n} a_{ik} x_k \ . \ (i = 1, 2, \cdots m)$$

Let $I = I(x_1, x_2, \cdots, x_n) = \|r\|_2^2 = \sum_{i=1}^{m} (b_i - \sum_{k=1}^{n} a_{ik} x_k)^2$, from the necessary condition for extreme value of multivariate function, we have

$$\frac{\partial I}{\partial x_j} = -2\sum_{i=1}^{m} (b_i - \sum_{k=1}^{n} a_{ik} x_k) a_{ij} = 0 , \quad (j = 1, 2, \cdots n).$$

It is

$$\sum_{k=1}^{n} (\sum_{i=1}^{m} a_{ij} a_{ik}) x_k = \sum_{i=1}^{m} a_{ij} y_i , \quad (j = 1, 2, \cdots n) \ .$$

This can be written compactly as

$$A^T Ax = A^T b .$$

Remark 1: *Theorem 3.1 shows that if we want to find the solution of equation $Ax = b$, we just have to find the solution of its normal equation $A^T Ax = A^T b$. According to this*

theorem, in order to find the solution of equation (2), this paper changes to find the solution of its normal equation (15).

Theorem 3.2: *Given equation $Ax = b$, if coefficient matrix A is $n \times n$, and positive definite, then the conjugate gradient algorithm will find the exact solution in n steps at most (Hu, 2008).*

Proof: Suppose the algorithm has operated $n-1$ steps, but still not find the solution vector, then the non-zero residual $r^{(0)}, \cdots, r^{(n-1)}$ form a group of orthogonal basis of R^n. In step n, there exist $r^{(n)} \perp r^{(i)}$, $i = 0, \cdots, n-1$, it is to say that $r^{(n)}$ is orthogonal with a group of basis of R^n, then $r^{(n)} = 0$, $e^{(n)} = 0$. So we have $x^{(n)} = x$.

Remark 2: *Theorem 3.2 shows that for exact linear searching conjugate gradient method, the dimensionality of searching space will reduce one-dimensional until reduce to zero and ends because each new searching direction is orthogonal with all previous directions. So this theorem provides the guarantee for the fast speed of conjugate gradient algorithm.*

3.2 CG-ELM

In view of the defect of SVD and advantages of conjugate gradient algorithm, conjugate gradient based ELM (CG-ELM) will be proposed based on theorems 3.1 and 3.2. This algorithm also serves for single-hidden layer feed-forward neural networks, randomly chooses the input weights and biases in hidden layer, but the output weights are determined by conjugate gradient algorithm. The algorithm can be summarised as follows:

Algorithm CG-ELM:

Input: Training set $X = \{(x_j, t_j) | x_j \in R^n, t_j \in R^m\}$, $j = 1, 2, \cdots, N$, activation function $g(x)$, hidden node number $\widetilde{N}$, standard error ε.

Output: Output weights of SLFNs.

Step 1: Randomly assign input weights ω_i and biases b_i, $i = 1, 2, \cdots, \widetilde{N}$;

Step 2: Calculate the hidden layer output matrix H;

Step 3: Calculate the output weights β of normal function $H^T H \beta = H^T T$ by using conjugate gradient algorithm;

Step 4: Set the output weights of original system $\widehat{\beta}$ by β calculated in step 3.

Remark 3: *From description above we can see, the only difference between ELM and CG-ELM is in step 3. In ELM, Moore-Penrose generalised inverse is used to find the output weights but conjugate gradient algorithm is used in CG-ELM.*

3.3 Computational complexity of CG-ELM

- Time complexity: From conjugate gradient algorithm we know the computations in this algorithm are mainly matrix addition and subtraction, whose time complexity is $O(n)$.
- Space complexity: Theorem 3.2 tells us, conjugate gradient algorithm will find the exact solution in n steps at most, which guarantees the storage space requirement in CG-ELM is acceptable and suitable for large scale data.

Through analysis above we can see that, compared with SVD, conjugate gradient algorithm has less time complexity and space complexity and is more suitable for large data. Simulation results in Section 4 also prove its advantage in speed.

4 Performance evaluation

In this section, the performance of the proposed CG-ELM is compared with the ELM proposed by Huang et al. (2006) on 30 data sets from UCI, in which 10 data sets are regression problems and the other 20 are classification problems. All the simulations are carried out in MATLAB 7.1 environment running in a founder (4 kernel, 3.1 GHz, 4GB memory, Windows 7 operating system). For the sake of simplicity, the activation functions used in these two algorithms are all sigmoid function $g(x) = 1/(1 + \exp(-x))$, and there are 20 hidden layer nodes assigned for ELM algorithm and CG-ELM algorithm.

4.1 Benchmarking with regression problems

The performances of ELM and CG-ELM are compared on 10 benchmark data sets of regression problem from UCI database, in which the sixth data set is an extreme large scale data (more than 40,000 samples). The specifications of these data sets are listed in Table 1.

Table 1 Specifications of regression data sets

No.	*Data sets*	*Samples*	*Attributes*
1	Airfoil Self-Noise	1503	5
2	Concrete Compressive Strength	1030	8
3	Computer Hardware	209	7
4	Forest Fires	517	8
5	Housing	506	13
6	Physicochemical Properties of PTS	45,730	9
7	Servo	62	3
8	Sinc	10,000	2
9	Wine Quality-red	1599	11
10	Wine Quality-white	4898	11

In our experiments, all attributes (both condition and decision attributes) have been normalised into the range [0, 1]. Ten-fold-cross-validation is used in all 10 regression data, which randomly divides data into 10 parts, nine parts as training set and one part as testing set. This process is conducted 10 times and the average result of 100 trials is taken as the final result and shown in Table 2 and Table 3. Evaluation standard in the regression problem includes training time, training accuracy, testing time and testing accuracy. The accuracy here is the root mean square error (RMSE). Training time ratio (TTR) of ELM and CG-ELM is shown in boldface if it is larger than 1.5 for a case.

Table 2 Comparison of training and testing RMSE of ELM and CG-ELM

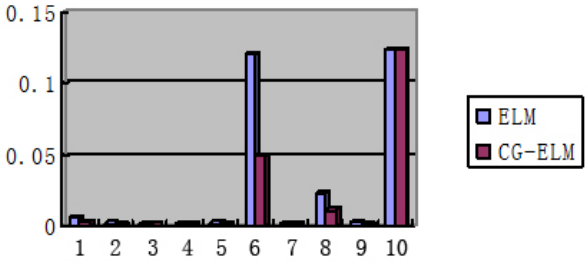
Data sets	*ELM*		*CG-ELM*	
	Training	*Testing*	*Training*	*Testing*
1	0.0351	181.9091	0.0384	1.5323
2	0.1137	0.1169	0.1138	0.1172
3	0.0033	0.0111	0.0045	0.0104
4	0.0668	0.1076	0.0699	0.0957
5	0.1194	0.2633	0.1218	0.2182
6	0.2417	0.2419	0.2426	0.2427
7	0.1377	0.1662	0.1377	0.1709
8	0.0820	0.0821	0.0831	0.0831
9	0.1276	0.1296	0.1277	0.1296
10	0.1235	0.1243	0.1236	0.1244

Table 3 Comparison of training and testing time of ELM and CG-ELM

Data sets	*ELM*		*CG-ELM*		*TTR*
	Training	*Testing*	*Training*	*Testing*	
1	0.0058	0.0003	0.0034	0.0002	**1.7059**
2	0.0022	0.0000	0.0012	0.0002	**1.8333**
3	0.0009	0.0002	0.0006	0.0002	**1.5000**
4	0.0014	0.0003	0.0011	0.0000	1.2727
5	0.0022	0.0005	0.0016	0.0003	1.3750
6	0.1204	0.0056	0.0498	0.0058	**2.4176**
7	0.0011	0.0000	0.0008	0.0000	1.3750
8	0.0236	0.0011	0.0115	0.0008	**2.0521**
9	0.0030	0.0005	0.0019	0.0000	**1.5789**
10	0.1235	0.1243	0.1236	0.1244	0.9992

As observed from Table 2, generally speaking, ELM and CG-ELM obtain similar generalisation performance. As shown in Table 3, CG-ELM obtains the faster training speed than ELM in almost all cases (except data 10, but on which almost the same speed). On six of ten data, CG-ELM runs 1.5 times faster than ELM, and the average ratio is 1.611, which demonstrates the advantage of the CG-ELM on training time. The training time comparison of two algorithms is shown in Figure 2.

Figure 2 The training time comparison of ELM and CG-ELM



4.2 Benchmarking with small and medium classification problems

The performance of ELM and CG-ELM are compared on 14 benchmark data sets of classification problems from UCI database, the specifications of which are listed in Table 4.

Table 4 Specifications of classification data sets

No.	*Data sets*	*Samples*	*Attributes*	*Classes*
11	Blood Transfusion	748	4	2
12	Breast Cancer	699	10	2
13	Breast Cancer W-P	198	33	2
14	Credit Approval	690	7	2
15	E. Coli Genes	327	5	5
16	Haberman's Survival	306	3	2
17	Heart Disease	270	13	2
18	Image Segmentation	2310	19	7
19	Libras Movement	360	90	15
20	Magic Telese (10%)	19,020 (10%)	10	2
21	Parkinsons	195	22	2
22	Page Blocks	5473	10	5
23	Sonar	208	60	2
24	Wine	178	13	3

Table 5 Comparison of training and testing correct classification rate of ELM and CG-ELM

Data sets	*ELM*		*CG-ELM*	
	Training	*Testing*	*Training*	*Testing*
11	0.7703	0.7890	0.7946	0.7861
12	0.9674	0.9630	0.9685	0.9633
13	0.8213	0.7707	0.8208	0.7697
14	0.7670	0.7472	0.7652	0.7504
15	0.9009	0.8892	0.8712	0.8693
16	0.7793	0.7472	0.7740	0.7463
17	0.8443	0.8137	0.8487	0.8137
18	0.8828	0.8821	0.8629	0.8587
19	0.7003	0.6156	0.6948	0.6183
20	0.8217	0.8169	0.8217	0.8170
21	0.8827	0.8517	0.8818	0.8532
22	0.9400	0.9220	0.9239	0.9174
23	0.7714	0.7153	0.7709	0.7143
24	0.9911	0.9719	0.9842	0.9645

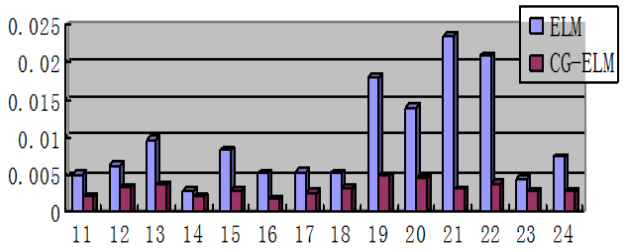
Just similar to regression case, in our classification experiments, all condition attributes have been normalised into the range [0,1]. Ten-fold-cross-validation is also used in all 14 classification data, and the average result of 100 trials is taken as the final result. All results are shown in Table 5 and Table 6. Evaluation standard in the classification problem also includes training time, training accuracy, testing time and testing accuracy. The accuracy here is the correct classification rate (CCR). Training time ratio (TTR) of ELM and CG-ELM is shown in boldface if it is larger than 2.0 for a case.

Table 6 Comparison of training and testing time of ELM and CG-ELM

Data sets	*ELM*		*CG-ELM*		*TTR*
	Training	*Testing*	*Training*	*Testing*	
11	0.0052	0.0000	0.0022	0.0003	**2.3636**
12	0.0063	0.0000	0.0034	0.0004	1.8529
13	0.0098	0.0000	0.0038	0.0003	**2.5789**
14	0.0030	0.0004	0.0022	0.0008	1.3636
15	0.0083	0.0005	0.0030	0.0000	**2.7667**
16	0.0053	0.0000	0.0019	0.0006	**2.7894**
17	0.0055	0.0003	0.0027	0.0005	**2.0370**
18	0.0053	0.0028	0.0033	**0.0013**	1.6060
19	0.0180	0.0019	0.0050	0.0006	**3.6000**
20	0.0141	0.0009	0.0047	0.0003	**3.0000**
21	0.0236	0.0000	0.0031	0.0006	**7.6129**
22	0.0209	0.0000	0.0039	0.0006	**5.3589**
23	0.0045	0.0016	0.0028	0.0001	1.6071
24	0.0075	0.0003	0.0028	0.0000	**2.6785**

As observed from Table 5, in these classification problems, ELM and CG-ELM also obtain similar generalisation performance. Similar as in regression problem, as shown in Table 6, CG-ELM also obtains the faster training speed than ELM in all cases. On ten of fourteen data, CG-ELM runs more than two times faster than ELM, and the average ratio is 2.944, which is much better than that in regression problems and shows more obvious advantage on training time. The training time comparison of two algorithms is shown in Figure 3.

Figure 3 The training time comparison of ELM and CG-ELM



4.3 Benchmarking with very large classification problems

The performance of ELM and CG-ELM are compared on 6 very large benchmark data sets of classification problems from UCI database, the specifications are listed in Table 7.

Table 7 Specifications of large classification data sets

No.	*Data sets*	*Samples*	*Attributes*	*Classes*
25	Adult	48842	14	5
26	Artificial-2State	250000	10	2
27	Cod-rna	488565	8	23
28	KDD Cup 1999 Data	4000000	14(35%)	5
29	MiniBooNE particle identification	130064	50	2
30	Skin-segmentation	245057	3	2

In this large scale classification experiments, we also normalised all condition attributes into [0,1]. Considering about the time, ten-fold-cross-validation is not used. For each case, the data are divided into 10 parts and nine parts as training data, one part as testing data before each trial of simulation. 10 trial have been conducted for the two algorithms and the average results are shown in Table 8 and Table 9.

Table 8 Comparison of training and testing correct classification rate of ELM and CG-ELM

Data sets	*ELM*		*CG-ELM*	
	Training	*Testing*	*Training*	*Testing*
25	0.7671	0.7628	0.7774	0.7721
26	0.7186	0.7192	0.7183	0.7193
27	0.9497	0.9482	0.9421	0.9410
28	–	–	0.9842	0.9512
29	0.8566	0.8569	0.8505	0.8496
30	0.8218	0.821	0.8212	0.8206

Table 9 Comparison of training and testing time of ELM and CG-ELM

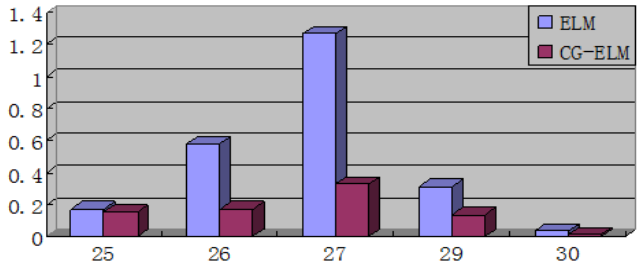
Data sets	*ELM*		*CG-ELM*		TTR
	Training	*Testing*	*Training*	*Testing*	
25	0.1716	0.0468	0.1560	0.0468	1.1000
26	0.5772	0.0780	0.1716	0.0780	3.3636
27	1.2636	0.1716	0.3276	0.1560	3.8571
28	–	–	7.8000	0.6552	–
29	0.3120	0.0624	0.1404	0.0624	5.0000
30	0.0372	0.0050	0.0154	0.0022	2.4156

In Table 9, "–" indicates null, because the ELM algorithm cannot run on this data.

As observed from Table 8, in the large scale classification problems, ELM and CG-ELM also obtain similar generalisation performance except on data 28, because ELM cannot run on data 28. We did try to run ELM for this application; however, it always ran out of memory in our ordinary PC. On the other hand, CG-ELM not only can handle this application, but also gets satisfactory results (training accuracy 98.42% and testing accuracy 95.12%). This application illustrates that the computational complexity of ELM is too high because of the computation of SVD, even leads to cannot running on some extreme large scale applications.

As shown in Table 9, CG-ELM also obtains the faster training speed than ELM in all cases. And on five of six data, CG-ELM has more than two times faster than ELM, and the average ratio is 3.1473 (without considering that ELM cannot run in data 28), which is even a little much better than that in small and medium problems and shows more obvious advantage on training time. The training time comparison of two algorithms is shown in Figure 4.

Figure 4 The training time comparison of ELM and CG-ELM



Considering we only replace Moore-Penrose generalised inverse with conjugate gradient method and have faster training time, we have reason to believe that it is the random assigning of weights and biases leads to the fast training speed of ELM. The above experiments also show that we can replace Moore-Penrose generalised inverse with conjugate gradient method to accelerate the training time. This strategy is especially suitable on large scale data because CG-ELM has good performance on some application that ELM cannot run.

4.4 Stability comparison of two algorithms

In this section, we will compare the stability of ELM and CG-ELM by Monte Carlo method. Idea of this method is: firstly create training data obey to specified distribution and train ELM and CG-ELM networks; secondly create a lot of simulation data, whose distribution just as the training data, put them into ELM and CG-ELM networks and compute the outputs; finally compute and compare the average variance of simulation outputs. The network which has the smaller variance has the better stability. That is because variance is used to measure the deviation of a random variable from its mathematical expectation. Greater variance means more dispersed distribution and worse stability, and smaller variance means more concentrated distribution and better stability.

This algorithm is described as follows:

Step 1: Create data sets A and $D_1, D_2, \cdots, D_{10}$ respectively as the training data and simulation data, which subject to the same distribution;

Step 2: Put the training data A into the two networks and get the trained ELM network model and CG-ELM network model;

Step 3: Put the simulation data $D_1, D_2, \cdots, D_{10}$ into two networks and compute the outputs;

Step 4: Compute the average variances of ten times simulation outputs as the standard of stability.

The training data includes 10000 samples, input data includes 5 conditional attributes, which obey standard normal distribution, and output data obey to uniform distribution on interval [0,1], [0,10], [0,100], [–1,1], [–10,10] and [–100,100] respectively. The change of the output interval is to make the variance more obvious to observe.

The simulation data only includes input data, which are 10000 samples obey standard normal distribution. This process is repeated 10 times, and the average variance is set as the standard to compare the stability of two networks. Table 10 lists the average variances of simulation outputs in six different training output data. As observed from Table 10, variances of simulation outputs in two algorithms are almost equal, which means that these two algorithms have the similar stability.

Table 10 Stability comparison of ELM and CG-ELM

Training output interval	*Average variance of simulation outputs*	
	ELM	*CG-ELM*
[0,1]	0.03017	0.03022
[0,10]	0.19394	0.19332
[0,100]	2.70117	2.70134
[–1,1]	0.03032	0.03030
[–10,10]	0.22231	0.22226
[–100,100]	2.83873	2.83737

5 Discussions and conclusion

In this paper we analysed the algorithm of ELM and pointed out it is random assignment of input weights and bias in hidden layer rather than the analytically computation of output weights by Moore-Penrose generalisation inverse leads to the fast training speed of ELM. In fact SVD, which is used to obtain Moore-Penrose generalised inverse, takes a lot of memory space and has high calculation complexity. Especially for extreme large amounts of data, this method is difficult to give a satisfactory result. Considering the conjugate gradient method not only has simple procedure but also can terminate in finite steps, conjugate gradient based ELM (CG-ELM) was proposed. On the premise that two algorithms have similar generalisation capability and stability, the proposed CG-ELM has faster speed than ELM. This demonstrates that it is the random weights assignment rather than the Moore-Penrose generalised inverse that leads to the fast training speed of ELM from some aspect. Less memory space, faster training speed and convenient incremental implementation make CG-ELM have great potential for the establishment and analysis for large scale data.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Project no. 71371063, 61672205,

61503252), The Science & Technology Bureau of Shenzhen (JCYJ20150324140036825), the Natural Science Foundation of Hebei Province (Project no. A2015210103), Youth Foundation of Hebei Province Department of Education Fund (Project no. QN2016140), and China Postdoctoral Science Foundation (2016T90799).

References

Bartlett, P.L. (1998) 'The sample complexity if pattern classification with neural networks: the size of the weights is more important than the size of the network', *IEEE Trans. Inf. Theory*, Vol. 44, No. 2, pp.525–536.

Cao, F., Wang, D., Zhu, H. and Wang, Y. (2016) 'An iterative learning algorithm for feedforward neural networks with random weights', *Information Science*, Vol. 1, No. 9, pp.546–557.

Dai, L. and Chen, X. (2016) 'Design of online water quality monitoring system and prediction based on probabilistic neural network', *International Journal of Wireless and Mobile Computing*, Vol. 10, No. 4, pp.371–377.

Ding, S., Zhao, H., Xu, X. and Nie, R. (2015) 'Extreme learning machine: algorithm, theory and applications', *Artificial Intelligence Review*, Vol. 44, No. 1, pp.103–115.

Emilio, S.O., Juan, G.S., Martin, J.D. et al. (2011) 'BELM: Bayesian extreme learning machine', *IEEE Transactions on Neural Networks*, Vol. 22, No. 3, pp.505–509.

Grippo, L., Lampariello, F. and Lucidi, S. (1986) 'A nonmonotone line search technique for Newton's method', *Siam Journal on Numerical Analysis*, Vol. 23, No. 4, pp.707–716.

Guliyev, N.J. and Ismailov, V.E. (2016) 'A single hidden layer feedforward network with only one neuron in the hidden layer can approximate any univariate function', *Neural Computation*, Vol. 28, No. 7, pp.1289–1304.

Guo, Z.-L., Huang, H.-X., Yang, H.-G., Wang, S.-W. and Wang, H. (2016) 'An enhanced gravitational search algorithm for global optimization', *International Journal of Wireless and Mobile Computing*, Vol. 10, No. 2, pp.183–190.

Hestenes, M.R. and Stiefel, E.L. (1952) 'Methods of conjugate gradients for solving linear systems', *Journal of Research National Bureau Standards*, Vol. 49, No. 6, pp.409–436.

Hu, M.L. (2008) *Matrix Calculation and Application*, Science Press, Beijing.

Huang, G.-B., Zhou, H.M., Ding, X.J. and Zhang, R. (2012) 'Extreme learning machine for regression and multi-class classification', *IEEE Transaction on System, Man, and Cybernetics-Part B: Cybernetics*, Vol. 42, No. 2, pp.513–529.

Huang, G.B., Zhu, Q.Y. and Siew, C.K. (2004) 'Extreme learning machine: a new learning scheme of feedforward neural networks', *Proceedings of International Joint Conference on Neural Networks*, Vol. 2, pp.985–990.

Huang, G.-B., Zhu, Q.-Y. and Siew, C.K. (2006) 'Extreme learning machine: theory and applications', *Neurocomputing*, Vol. 70, Nos. 1–3, pp.489–501.

Jamli, M.R., Ariffin, A.K. and Wahab, D.A. (2015) 'Incorporating feedforward neural network within finite element analysis for L-bending springback prediction', *Expert Systems with Applications*. Vol. 42, No. 5, pp.2604–2614.

Kwok, T.Y. and Yeung, D.Y. (1997) 'Constructive algorithm for structure learning feedforward neural networks for regression problems', *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, pp.630–645.

Liang, N.Y., Huang, G.-B., Saratchandran, P. and Sundararajan, N. (2006) 'A fast and accurate online sequential algorithm for feedforward networks', *IEEE Transactions on Neural Networks*, Vol. 17, No. 6, pp.1411–1423.

Liu, T., Hu, L., Ma, C. et al. (2015) 'A fast approach for detection of erythemato-squamous diseases based on extreme learning machine with maximum relevance minimum redundancy feature selection', *International Journal of Systems Science*, Vol. 46, No. 5, pp.919–931.

Marquardt, D.W. (1963) 'An algorithm for least-squares estimation of nonlinear parameters', *SIAM Journal on Applied Mathematics*, Vol. 11, pp.431–441.

Mason, L., Baxter, J., Bartlett, P. and Frean, M. (1999) 'Boosting algorithms as gradient descent', *Discrete Mathematics*, Vol. 30, No. 3, pp.303–304.

Mohammed, A.A., Minhas, R., Jonathan, Q.M. et al. (2011) 'Human face recognition based in multidimensional PCA and extreme learning machine', *Pattern Recognition*, Vol. 44, Nos. 10/11, pp.2588–2597.

Qiu, J.-L., Zhang, L.-L., Fan, T.-H., Wang, Y. and Wang, H.-B. (2016) 'Data fusion algorithm of multilayer neural network by ZigBee Protocol architecture', *International Journal of Wireless and Mobile Computing*, Vol. 10, No. 3, pp.214–223.

Wang, J., Cai, Q., Chang, Q. and Zurada, J.M. (2016) 'Convergence analyses on sparse feedforward neural networks via group lasso regularization', *Information Sciences*, Vol. 381, pp.250–269.

Wang, M.-P. and Tian, Q. (2015) 'Prediction of heating parameters based on support vector machine', *International Journal of Wireless and Mobile Computing*, Vol. 8, No. 3, pp.294–300.

Werbos, P.J. (1994) *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, John Wiley & Sons, New York, NY, USA.

Wu, J., Wang, S.T. and Chung, F.L. (2011) 'Positive and negative fuzzy rule system, extreme learning machine and image classification', *International Journal of Machine Learning and Cybernetics*, Vol. 2, No. 4, pp.261–271.

Zhang, L. and Suganthan, P.N. (2016) 'A survey of randomized algorithms for training neural networks', *Information Science*, Vol. 364, pp.146–155.

Zhao, S.-X. and Wang, X.-Z. (2014) 'Extreme learning machine for interval-valued data', *Proceeding of 13th International Conference on Machine Learning and Cybernetics*, pp.388–399.

Zhu, Q.Y., Qin, A.K., Suganthan, P.N. et al. (2005) 'Evolutionary extreme learning machine', *Pattern Recognition*, Vol. 38, No. 10, pp.1759–1763.