An Improved Extreme Learning Machine Based on Full Rank Cholesky Factorization

Zuozhi Liu1,a, JinJian Wu2 and Jianpeng Wang3
1School of Mathematics & Statistics, Guizhou University of Finance and Economics, Guiyang, Guizhou, China
2School of Artificial Intelligence, Xidian University, Xi’an, Shaanxi, China
3Department of Mathematics and Physics, Changzhou Campus, Hohai University, Changzhou, Jiangsu, China

Abstract. Extreme learning machine (ELM) is a new novel learning algorithm for generalized single-hidden layer feedforward networks (SLFNs). Although it shows fast learning speed in many areas, there is still room for improvement in computational cost. To address this issue, this paper proposes an improved ELM (FRCF-ELM) which employs the full rank Cholesky factorization to compute output weights instead of traditional SVD. In addition, this paper proves in theory that the proposed FRCF-ELM has lower computational complexity. Experimental results over some benchmark applications indicate that the proposed FRCF-ELM learns faster than original ELM while preserving good generalization performance.

1 Introduction

As a universal approximator, the single-hidden-layer feedforward networks (SLFNs) have been widely studied and used in different fields in the past decades [1-3]. However, the slow traditional learning algorithms cannot satisfy the need of the rapid development of SLFNs. To address issue, Huang et al. have proposed a simple and efficient learning algorithm for SLFNs named extreme learning machine (ELM) [4]. This technique randomly chooses the parameters (input weights and biases) of hidden nodes and analytically determines the output weights according to least-mean square method [5]. Compared to the traditional gradient-based learning algorithms, ELM not only provides better generalization performance with fast learning speed, but also avoids many issues faced by traditional algorithms such as stopping criteria, learning rate, local minima and overfitting. In view of these advantages, ELM has been widely studied and applied in various areas [6-9].

Although ELM shows fast learning speed in many areas, there is still room for improvement in computational complexity. By analyzing the theory of ELM, we can find that the learning time of ELM is mainly consumed by computing the Moore-Penrose generalized inverse of hidden layer output matrix (denoted by $H^\dagger$). Technically, the computation of $H^\dagger$ is generally completed based on the singular value decomposition (SVD) and the corresponding computational complexity is $O(4NL + 8L^3)$ [10]. It can be seen that when the data size $N$ or the number of hidden neurons $L$ increases, the necessary computational complexity will yet become large. Therefore, the large complexity of computing $H^\dagger$ is the key factor restricting the learning speed of ELM.

To address this problem, this paper proposes an improved ELM based on the full rank Cholesky factorization, named FRCF-ELM. Different from the original ELM, FRCF-ELM computes the output weights by using the simpler full rank Cholesky factorization instead of the traditional SVD. Furthermore, we systematically analyze the computational complexity of such FRCF-ELM in theory. It will be verified that the obtained FRCF-ELM has lower computational complexity. Experimental results over some benchmark applications also indicate that the proposed FRCF-ELM learns faster than the original ELM while preserving good generalization performance.

2 Review of ELM

In this section, we briefly describe the ELM proposed by Huang et al.. For a given set of training examples $\{(x_i, t_i)\}_{i=1}^N \subset R^r \times R^d$, if the outputs of standard SLFNs are equal to the targets, we can obtain

$$f_\beta(x) = \sum_{i=1}^{N} \beta_i G(a_i, b_i, x) = t, j = 1, \ldots, N$$

(1)

where $\beta_i = [\beta_{i1}, \ldots, \beta_{iN}]$ is the weight vector connecting the $i$th hidden node and the output nodes, $a_i$ is the weight vector connecting the input layer to the $i$th hidden node, $b_i$ is the bias of the $i$th hidden node, $G(a_i, b_i, x)$ is the output of the $i$th hidden node with respect to the input $x$.

The above $N$ equations can be written compactly as

$$H\beta = T$$

(2)

where
In ELM, the parameters (input weights \( \mathbf{a} \) and biases \( b \)) of output matrix \( \mathbf{H} \) are randomly generated. Thus, training the SLFNs is simply equivalent to finding the solution of the linear system (2) with respect to the output weight \( \mathbf{β} \). In fact, the output weight \( \mathbf{β} \) is determined to be the minimum norm least squares solution of (2), that is

\[
\hat{\mathbf{β}} = \mathbf{H}^\dagger \mathbf{T} \tag{3}
\]

where \( \mathbf{H}^\dagger \) is the Moore-Penrose generalized inverse of the output matrix \( \mathbf{H} \). It should be noted that ELM computes \( \mathbf{H}^\dagger \) based on the singular value decomposition (SVD) of \( \mathbf{H} \), that is

\[
\mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{\sigma}_i \mathbf{v}_i^T \tag{4}
\]

\[
\mathbf{H}^\dagger = \mathbf{V} \Sigma^\dagger \mathbf{U}^T = \sum_{i=1}^{r} v_i^T \mathbf{u}_i / \mathbf{\sigma}_i \tag{5}
\]

Generally speaking, ELM learning algorithm can be summarized as follows:

**ELM Algorithm:** Given a set of training examples \( \{(\mathbf{x}_i,t_i)\}_{i=1}^{n} \subseteq \mathbb{R}^r \times \mathbb{R}^o \), activation function \( G(\cdot) \), and hidden node number \( L \),

1. Randomly assign input weight \( \mathbf{a}_i \) and bias \( b_i \), \( i = 1, \ldots, L \).
2. Calculate the hidden layer output matrix \( \mathbf{H} \).
3. Calculate the output weight \( \mathbf{β} : \mathbf{β} = \mathbf{H}^\dagger \mathbf{T} \).

### 3 Proposed FRCF-ELM

In this section, we propose an improved ELM based on the full rank Cholesky factorization, named FRCF-ELM, which aims to further improve the learning speed of ELM. Different from the original ELM, the core idea of the proposed FRCF-ELM is to compute the output weights by using the simpler full rank Cholesky factorization instead of traditional SVD.

#### 3.1. Formulation of FRCF-ELM

According to the property of the Moore-Penrose inverse \([10,11]\), the Moore-Penrose inverse of a matrix product \( \mathbf{A} \mathbf{B} \) is as follows

\[
(\mathbf{A} \mathbf{B})^\dagger = (\mathbf{B}^\dagger (\mathbf{A}^\dagger \mathbf{A} \mathbf{B}^\dagger))^\dagger \mathbf{A}^\dagger \tag{6}
\]

With \( \mathbf{A} = \mathbf{H} \), \( \mathbf{B} = \mathbf{I} \), the Moore-Penrose inverse of output matrix \( \mathbf{H} \) can be formulated as

\[
\mathbf{H}^\dagger = (\mathbf{H}^\dagger \mathbf{H})^\dagger \mathbf{H}^\dagger \tag{7}
\]

Note that \( \mathbf{H}^\dagger \mathbf{H} \) is a symmetric positive \( L \times L \) matrix. Here, we set the rank of \( \mathbf{H}^\dagger \mathbf{H} \) is \( r (r \leq L) \). According to the theory of full rank Cholesky factorization \([12]\), there is a unique upper triangular matrix \( \mathbf{S} \in \mathbb{R}^{r \times r} \) such that

\[
\mathbf{H}^\dagger \mathbf{H} = \mathbf{S}^\dagger \mathbf{S} \tag{8}
\]

If \( \mathbf{A} = \mathbf{S}^\dagger \), \( \mathbf{B} = \mathbf{S} \), then one obtains from (6)

\[
(\mathbf{S}^\dagger \mathbf{S})^\dagger = (\mathbf{S}^\dagger (\mathbf{S} \mathbf{S}^\dagger))^\dagger \mathbf{S} = \mathbf{S}^\dagger (\mathbf{S}^\dagger)^\dagger (\mathbf{S}^\dagger)^\dagger \mathbf{S} \tag{9}
\]

Using Eqs. (8) and (9), one obtains from (7)

\[
\mathbf{H}^\dagger = (\mathbf{H}^\dagger \mathbf{H})^\dagger \mathbf{H}^\dagger = (\mathbf{S}^\dagger \mathbf{S})^\dagger \mathbf{H}^\dagger = \mathbf{S}^\dagger (\mathbf{S} \mathbf{S}^\dagger)^\dagger (\mathbf{S}^\dagger)^\dagger \mathbf{S}^\dagger \mathbf{T} \tag{10}
\]

As described above, we can obtain a new method for the computation of Moore-Penrose inverse matrices. By introducing this method into ELM, we can obtain an improved ELM named FRCF-ELM. Simply speaking, FRCF-ELM computes the output weights using the simpler full rank Cholesky factorization instead of traditional SVD. FRCF-ELM can be summarized as follows:

**FRCF-ELM Algorithm:** Given a set of training data \( \{(\mathbf{x}_i,t_i)\}_{i=1}^{n} \subseteq \mathbb{R}^r \times \mathbb{R}^o \), activation function \( G(\cdot) \), and hidden node number \( L \),

1. Randomly assign input weight \( \mathbf{a}_i \) and bias \( b_i \), \( i = 1, \ldots, L \).
2. Compute the hidden layer output matrix \( \mathbf{H} \).
3. Implement full rank Cholesky factorization to \( \mathbf{H}^\dagger \mathbf{H} \) to obtain the upper triangular matrix \( \mathbf{S} \in \mathbb{R}^{r \times r} \) such that \( \mathbf{H}^\dagger \mathbf{H} = \mathbf{S}^\dagger \mathbf{S} \).
4. Calculate the output weight \( \mathbf{β} : \mathbf{β} = \mathbf{S}^\dagger (\mathbf{S} \mathbf{S}^\dagger)^\dagger (\mathbf{S}^\dagger)^\dagger \mathbf{S}^\dagger \mathbf{T} \tag{11} \)

### 3.2 Analysis of computational complexity

In ELM, the output weight \( \mathbf{β} \) is calculated based on SVD of the output matrix \( \mathbf{H} \in \mathbb{R}^{n \times L} \). Thus the computational complexity of ELM is approximately equal to that of SVD. As described in \([10]\), the computational complexity of applying SVD to \( \mathbf{H} \) is \( O(4NL' + 8L') \). Therefore, we can deem that the computational complexity of ELM is \( O(4NL' + 8L') \).

In the previous subsection, we can find that the complexity of the proposed FRCF-ELM is mainly composed of three aspects: full rank Cholesky factorization, inversion of matrix and matrix product. The computational complexities of these three operations are \( O(L'/3) \), \( O(2r'/3) \) and \( O(E(N+r)) \), respectively.
Therefore, the total computational complexity of the proposed FRCF-ELM is $O(L'/3 + 2r^3/3 + L'(N + r))$.

In order to better illustrate the superiority of FRCF-ELM in the learning speed, we then compare the computational complexities of FRCF-ELM and the original ELM,

$$\frac{O(\text{FRCF-ELM})}{O(\text{ELM})} = \frac{O(L'/3 + 2r^3/3 + L'(N + r))}{O(4NL' + 8L')}$$

and further derive that

$$\frac{L'/3 + 2r^3/3 + L'(N + r)}{4NL' + 8L'} = \frac{\frac{L}{3N} + \frac{2r^3}{3N} + \frac{L}{N} + \frac{r}{N} + 1}{4 + \frac{8L}{N}}$$

Because $r < L \ll N$, one can obtain that $L/N \rightarrow 0$, $r/N \rightarrow 0$ and $r/L \rightarrow 1$. Then Eq. (13) can be finally derived as

$$\frac{\frac{L}{3N} + \frac{2r^3}{3N} + \frac{r}{N} + 1}{4 + \frac{8L}{N}} < 1$$

Based on the above analyses, we can conclude that the computational complexity of FRCF-ELM is smaller than that of ELM. In other words, FRCF-ELM learns faster than ELM.

4 Experimental verification

In this section, we carried out a series of experiments to test the proposed FRCF-ELM. In order to better verify the effectiveness of our method, we compare it with ELM and popular support vector machine (SVM). Here, the SVM is performed using the latest LIBSVM software package (3.23 version) [13]. In addition, we select epsilon-SVR for regression problems and nu-SVC for classification problems.

We compare the performance of these three learning algorithms on some benchmark problems from UCI database [14], including six regression problems and three classification problems. The specification of these datasets is shown in Table 1. For each problem, the dataset is randomly divided into two subsets: a training set for model learning and a testing set for performance quantification. In order to make a fair comparison, 50 trials have been conducted and the median results are marked. All the simulations of different algorithms are running in MATLAB 2011a environment on the PC with Intel Core 3.40 GHz CPU and 4 GB RAM.

Tables 2 and 3 show the performance comparison of ELM, SVM and the proposed FRCF-ELM on regression problems and classification problems, respectively. The apparent better results are emphasized in boldface. From Tables 2 and 3, it can be seen that the proposed FRCF-ELM always has faster learning speed than two other methods for all nine problems. For example, in the case of Census (House8L) problem, FRCF-ELM learns up to 4 times faster than ELM and 12 times faster than SVM. In addition, it also can be seen that the generalization performance of FRCF-ELM is similar or comparable to ELM and SVM. These facts indicate that the proposed FRCF-ELM learns faster while preserving good generalization performance.

Table 1. Specification of benchmark datasets.

| Problems       | Attributes | Training Data | Testing Data |
|----------------|------------|---------------|--------------|
| Regression     |            |               |              |
| Abalone        | 8          | 2000          | 2177         |
| California Housing | 8          | 8000          | 12640        |
| Census (House8L) | 8          | 10000         | 12784        |
| Delta Elevators | 6          | 3000          | 4129         |
| Delta Elevators     | 6          | 4000          | 5517         |
| Japanese Vowels   | 12         | 4274          | 5687         |
| Classification   |            |               |              |
| Waveform        | 21         | 2500          | 2500         |
| Satellite Image | 36         | 4435          | 2000         |
| Letter          | 16         | 13333         | 6667         |

Table 2. Performance comparison of FRCF-ELM, ELM and SVM on regression problems.

| Problems       | Algorithms    | Training time (s) | Testing RMSE Mean | Dev |
|----------------|---------------|-------------------|--------------------|-----|
| Abalone        | FRCF-ELM     | 0.0133            | 0.0770             | 0.0013 |
|                | ELM           | 0.0250            | 0.0771             | 0.0015 |
|                | SVM           | 0.4688            | 0.0780             | 0.0012 |
| California     | FRCF-ELM     | 0.1914            | 0.1306             | 0.0037 |
| Housing        | ELM           | 0.7156            | 0.1312             | 0.0038 |
|                | SVM           | 10.3824           | 0.1328             | 0.0013 |
| Census (House8L)| FRCF-ELM   | 0.5844            | 0.0683             | 0.0071 |
|                | ELM           | 2.4414            | 0.0684             | 0.0092 |
|                | SVM           | 7.0612            | 0.0712             | 0.0013 |
| Delta Elevators| FRCF-ELM     | 0.0336            | 0.0391             | 0.0006 |
|                | ELM           | 0.0898            | 0.0391             | 0.0007 |
|                | SVM           | 0.8156            | 0.0392             | 0.0007 |
| Japanese Vowels| FRCF-ELM     | 0.2406            | 0.0590             | 0.0013 |
|                | ELM           | 0.9531            | 0.0591             | 0.0010 |
|                | SVM           | 3.0893            | 0.0595             | 0.0006 |

Table 3. Performance comparison of FRCF-ELM, ELM and SVM on classification problems.

| Problems       | Algorithms    | Training time (s) | Testing Accuracy Mean | Dev |
|----------------|---------------|-------------------|-----------------------|-----|
| Waveform       | FRCF-ELM     | 0.2344            | 85.56                 | 0.0068 |
|                | ELM           | 0.6906            | 85.51                 | 0.0072 |
|                | SVM           | 1.1674            | 85.42                 | 0.0054 |
| Satellite Image| FRCF-ELM     | 2.1445            | 89.55                 | 0.0050 |
|                | ELM           | 6.1289            | 89.42                 | 0.0040 |
|                | SVM           | 5.3760            | 89.60                 | 0.0000 |
| Letter         | FRCF-ELM     | 5.0344            | 89.48                 | 0.0033 |
|                | ELM           | 15.4187           | 89.35                 | 0.0039 |
|                | SVM           | 37.6487           | 90.09                 | 0.0052 |

5 Conclusion

In this paper, an enhanced ELM called FRCF-ELM is proposed which can greatly improve the learning speed of ELM. In FRCF-ELM, the output weights are computed using simpler full rank Cholesky factorization instead of
traditional SVD. It has been proved in theory that FRCF-ELM has lower computational complexity. Experimental results over some benchmark applications also indicate that the proposed FRCF-ELM learns faster than original ELM while preserving good generalization performance.

Acknowledgement

This study was supported by the Talents Introduction Research Projects of Guizhou University of Finance and Economics (2018YJ14) and the Fundamental Research Funds for the Central Universities (2018B24214).

References

1. M.T. Hagan, H.B. Demuth, M.H. Beale, *Neural network design* (Boston, MA: PWS-Kent, 1996)
2. S. Haykin, *Neural Networks: A comprehensive foundation* (NewYork: Macmillan, 1994)
3. C.G. Looney, *Pattern recognition using neural networks: theory and algorithms for engineers and Scientists* (New York: Oxford Univ. Press, 1997)
4. G.B. Huang, Q.Y. Zhu, C.K. Siew, Neurocomputing 70, 1-3, 489-501 (2006)
5. G.B. Huang, Q.Y. Zhu, K.Z. Mao, C.K. Siew, P. Saratchandran, N. Sundararajan, IEEE Trans. Circuits Syst. II Exp. Briefs 53, 3, 187-191 (2006)
6. R.X. Zhang, G.B. Huang, N. Sundararajan, P. Saratchandran, IEEE ACM Trans. Comput. Biol. 4, 3, 485-495 (2007)
7. G.R. Wang, Y. Zhao, D. Wang, Neurocomputing 72, 1, 262-268 (2008)
8. Q. Yuan, W.D. Zhou, S.F. Li, D.M. Cai, Epilepsy Res. 96, 1-2, 29-38, (2011)
9. W.W. Zong, G.B. Huang, Neurocomputing 74, 1, 2541-2551 (2011)
10. G.H. Golub, C.F.V. Loan, *Matrix computations, third ed.* (Maryland , Johns Hopkins University Press, 1996)
11. M.A. Rakha, Appl. Math. Comput. 158, 1, 185-200 (2004)
12. P. Courrieu, Comput. Sci. 8, 2, 25 (2008)
13. C.C. Chang, C.J. Lin, *LIBSVM: a library for support vector machines* (2001) <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
14. A. Asuncion, D.J. Newman, *UCI machine learning repository* (Irvine, CA, University of California, School of Information and Computer Science, 2007) <http://www.ics.uci.edu/mllearn/MLRepository.html>