A FAST LEARNING ALGORITHM FOR MULTI-LAYER EXTREME LEARNING MACHINE

Jiexiong Tang, Chenwei Deng *

School of Information and Electronics
Beijing Institute of Technology, China
Email:{cwdeng, jiexiongtang}@bit.edu.cn

Guang-Bin Huang, Junhui Hou

School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
Email: {egbhuang, houj0001}@ntu.edu.sg

ABSTRACT

Extreme learning machine (ELM) is an efficient training algorithm originally proposed for single-hidden layer feed-forward networks (SLFNs), of which the input weights are randomly chosen and need not to be fine-tuned. In this paper, we present a new stack architecture for ELM, to further improve the learning accuracy of ELM while maintaining its advantage of training speed. By exploiting the hidden information of ELM random feature space, a recovery-based training model is developed and incorporated into the proposed ELM stack architecture. Experimental results of the MNIST handwriting dataset demonstrate that the proposed algorithm achieves better and much faster convergence than the state-of-the-art ELM and deep learning methods.

Index Terms— Extreme learning machine (ELM), deep learning, multi-layer training, sparse representation.

1. INTRODUCTION

Extreme learning machine (ELM), proposed by Huang *et al.* [1], is a training scheme for single-hidden layer feed-forward network (SLFNs). Unlike the other training algorithms, e.g., neural networks (NNs), or support vector machine (SVM), the parameters of ELM hidden layer need not be adjusted, and could be independent of the training data. ELM has demonstrated to have much faster learning speed and lower computational complexity than NNs and SVM in numerous applications, such as face recognition [2], image segmentation [3], and human action recognition [4], etc.

Theoretically, ELM is able to approximate any complex nonlinear mappings directly from the training samples. However, similar to the traditional SLFNs, ELM is established by a shallow architecture [1]. Therefore, in order to perfectly fit the highly-variant input data, it may need a tremendous large network that would not be easy to implement.

*This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 61301090; and in part by the Excellent Young Scholars Research Fund of Beijing Institute of Technology under Grant 2013YR0508.

Inspired by the deep learning (DL) [5], a multi-layer ELM (ML-ELM) [6] was developed to address this issue. ML-ELM decomposes the original inputs into multiple hidden layers, the output of previous layer is used as the input of the current layer with an orthogonal initialization, and the weighting parameters of each hidden layer are set by layer-wise unsupervised training. However, when the number of input nodes is different from that of the output ones, the orthogonal constraint of the input weights cannot be satisfied for the least mean square training in [6].

Different from the existing ML-ELM and DL schemes, in this paper, we propose a new and fast stack architecture for ELM, to further improve the training speed of ML-ELM and DL, while achieving higher learning accuracy. By using ℓ_1 norm optimization of the output weights, more sparse hidden layers are generated. Moreover, ELM random feature space is utilized for better representation, and this helps to achieve more adaptive data mapping.

The rest of this paper is organized as follows. In Section 2, some basic principles of ELM are briefly introduced; in Section 3, the proposed stack architecture of ELM and its related optimization method are presented in detail; Experimental results and analysis are given in Section 4, and we compared the results of proposed scheme with those of state-of-the-art ELM and DL methods; while Section 5 concludes this paper.

2. BASIC PRINCIPLES OF ELM

ELM is a fast training algorithm originally developed for the SLFNs, and has been extented to the generalized SLFNs. In this section, we briefly introduce the basic ideas of ELM.

Given a training set $N = \{(x_i, t_i)|x_i \in R^n, t_i \in R^m, i = 1, \cdots, L\}$, where x_i is the training data vector, t_i represents the class label of each sample, and L denotes the number of hidden nodes. The ELM training algorithm can be summarized as follows [1]:

(1) Randomly assign the hidden node parameters, e.g., the input weights w_i and biases b_i, $i = 1, \cdots, L$.

(2) Calculate the hidden layer output matrix H.

(3) Obtain the output weight vector $\beta = H^{\dagger}T$, where $T =$

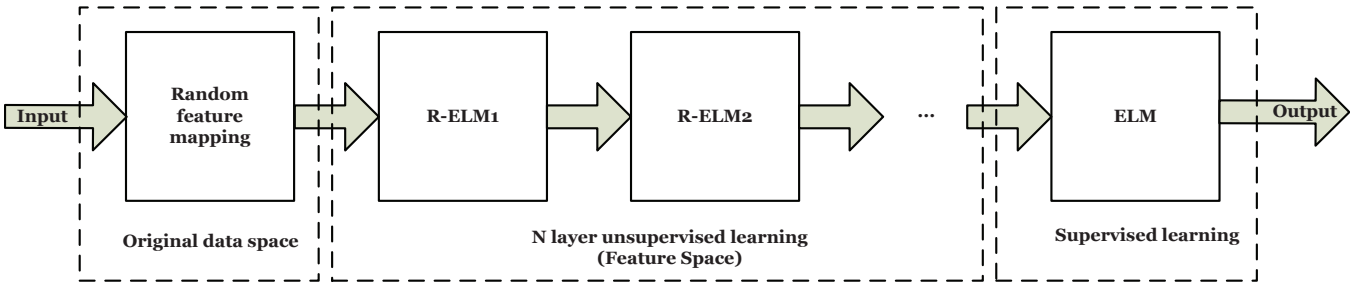

Fig. 1. Architecture of the proposed framework.

$[t_1, \cdots, t_L]^T$, $H^\dagger$ is the Moore-Penrose generalized inverse of matrix H.

The orthogonal projection method can be efficiently used for the calculation of MP inverse: $H^\dagger = (H^T H)^{-1} H^T$, if $H^T H$ is nonsingular; or $H^\dagger = H^T (H^T H)^{-1}$, if HH^T is nonsingular.

According to the ridge regression theory, it was suggested that a positive value $\frac{1}{\lambda}$ is added to the diagonal of $H^T H$ or HH^T in the calculation of the output weights β, and by doing so, the resultant solution is more stable and has better generalization performance [1]. That is, in order to improve the stability of ELM, we can have $\beta = H^T (\frac{1}{\lambda} + HH^T)^{-1} T$ (or $\beta = (\frac{1}{\lambda} + HH^T)^{-1} H^T T$), and the corresponding output function of ELM is $f(x) = h(x)\beta = h(x) H^T (\frac{1}{\lambda} + HH^T)^{-1} T$ (or $f(x) = h(x)\beta = h(x)(\frac{1}{\lambda} + HH^T)^{-1} H^T T$).

3. PROPOSED METHOD

3.1. ELM Stack Architecture

The proposed ELM stack architecture is built in a multi-layer manner, as is shown in Fig.1. Before training, the input data should be transformed to an ELM random feature space, which can help to exploit the hidden information among training samples. Then, a N-layer unsupervised learning is performed to eventually obtain the high-level sparse features. The output of the hidden layer can be represented as

$$H_i = g(H_{i-1} \times \beta) \tag{1}$$

where H_i is the output of the ith layer ($i \in [1, N]$), H_{i-1} is the output of $(i-1)$th layer and $g(\cdot)$ denotes the activation function of hidden layers. The output of Nth layer unsupervised learning H_N is used as the input of the supervised ELM training to obtain the final classification or regression result of the whole network.

In addition, it is known that adding "noise" to the original inputs can force the hidden weights to capture more important information [7]. Therefore, before unsupervised encoding, the input data will firstly scattered by a random matrix, which is similar to the corruption process in the stacked denoising autoencoder (SDA) of deep learning (DL). The corrupted data is then "recovered" by ℓ_1 optimization, and the reason for the use of ℓ_1 norm is that it can obtain more sparse feature representation of the input data and also have better performance for data recovery. Since the recovered data is finally utilized for the layer-wise unsupervised training in Fig.1, we term it as *recovery-based ELM* (R-ELM), and the details of R-ELM optimization are to be discussed in next subsection.

It should be noted that the proposed ELM stack architecture is different from the traditional DL [5]. From concept point of view, DL considers multi-layer as a whole with unsupervised initialization, and after such initialization the entire network will be trained by back propagation (BP) NNs, all layers are "hard coded" together. In contrast, each layer in the proposed ELM stack architecture can be considered as a separate part (or an autonomous sub-system/sub-module). It is something like feature extraction and then classification, and fine-tuning is not required for the entire system combined by feature extraction and classification, and thus the training speed can be much faster than traditional BP-based DL.

On the other hand, the proposed learning scheme differentiates itself from the existing ML-ELM [6] in twofold: 1) the orthogonal initialization is avoided, since the orthogonal constraint is not reasonable when the number of the input nodes is different from that of the output ones; 2) instead of ℓ_2 norm used in ML-ELM, ℓ_1 penalty is applied to generate more sparse and meaningful hidden features.

3.2. R-ELM Optimization

As introduced in previous subsection, the proposed learning framework briefly consists of two parts: unsupervised and supervised training. Since the supervised training is implemented by the original ELM, in this subsection, we will focus on how to train the unsupervised building blocks (i.e., R-ELM) of the proposed ELM stack architecture. The structure of R-ELM is shown in Fig.2.

It can be seen that unlike the traditional BP-based NNs, the input weights of R-ELM are established by searching the path back from the random space. The ELM theory in [8, 9] has proved that random feature mapping (with almost any

nonlinear activation function) can provide universal approximation capability, and by doing so, more important information can be exploited for hidden layer feature representation.

The optimization model of R-ELM can be denoted as the following equation.

$$O_\beta = \underset{\beta}{argmin} \left\{ ||H\beta - X||^2 + ||\beta||_{\ell_1} \right\} \tag{2}$$

where X denotes the input data, H is the output of random mapping and β is the hidden layer weights to be obtained.

In the existing DL algorithms, X is usually the encoding outputs of the bases β, which need to be adjusted during the iterations of optimization. However, in R-ELM, as we are to utilize random mapping for hidden layer feature representation, X is the original data and H is the random initialized output which need not to be optimized [8, 9]. Furthermore, the experiments in the next section will show that it would not only help to improve the training time but also the learning accuracy.

In the following, we will describe the optimization algorithm for the ℓ_1 norm recovery problem. For clear representation, we re-write the object function in Eq. 2 as

$$O_\beta = f(\beta) + g(\beta) \tag{3}$$

where $f(\beta) = ||H\beta - X||^2$, and $g(\beta) = ||\beta||_{\ell_1}$ is the ℓ_1 penalty term of the training model.

Then, a fast iterative shrinkage-thresholding algorithm (FISTA) [10] is adopted to efficiently solve the problem in Eq.3, by minimizing a smooth convex function with the complexity of $O(1/k^2)$, where k is the iteration times. An example of recovered data obtained by FISTA is depicted in Fig.3, it can be seen that the characteristics of original data are maintained in the recovered ones although the data in Fig. 3(b) is reconstructed from highly nonlinear transformation.

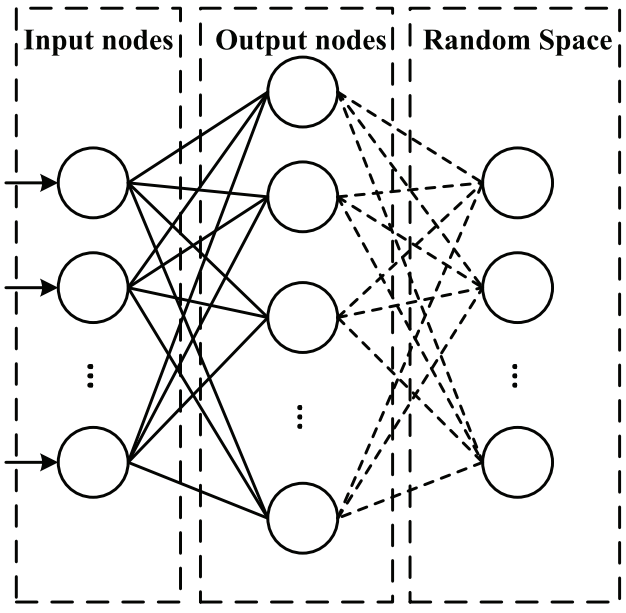

Fig. 2. R-ELM architecture.

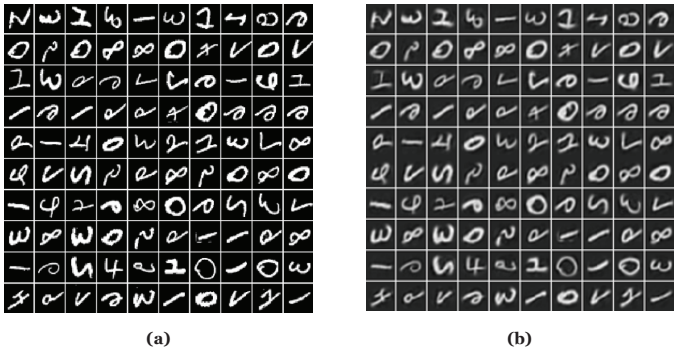

Fig. 3. Visualization of recovered data: (a) original dataset; (b) recovered dataset.

The implementation details of FISTA are as follows [10]:

(1) Calculate the Lipschitz constant of the gradient of smooth convex function ∇f.

(2) Begin the iteration by taking $y_1 = \beta_0 \in R^n$, $t_1 = 1$ as the initial points. Then for $k(k \geq 1)$:

(a) $\beta_k = p_L(y_k)$, where p_L is given by:

$p_L = \underset{\beta}{argmin} \left\{ \frac{L}{2} ||\beta - (\beta_{k-1} - \frac{1}{L}\nabla f(\beta_{k-1})||^2 + g(\beta) \right\}$

(b) $t_{k+1} = \frac{1+\sqrt{1+4t_k{}^2}}{2}$

(c) $y_{k+1} = \beta_k + (\frac{t_k - 1}{t_{k+1}})(\beta_k - \beta_{k-1})$,

By computing the iterative steps above, we could manage to recover the data from the corrupted ones. By using the resultant bases β as the weights of R-ELM, the correlation between the input and learned features would reflect the compact representations of the original data. And as R-ELM is adopted as the building block of proposed framework, higher level feature representations can be generated by layer-wise comparison. Also, compared with ML-ELM [6] which simply uses the input as output with traditional least mean square method, the ℓ_1 optimization has been proved [10] to be a better solution for data recovery and other applications. It will help to reduce the number of neural nodes and thus further improve the testing time of ELM.

4. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, the learning accuracy and training speed of proposed method are compared with the relevant state-of-the-art schemes. And the widely-used MNIST handwriting dataset [11] is used to compare the performance of proposed method with those of ML-ELM [6], DBN [5], and ELM [1]. This dataset consists of 60000 training images and 10000 testing images. In our simulations, the testing hardware and software conditions are listed as follows.

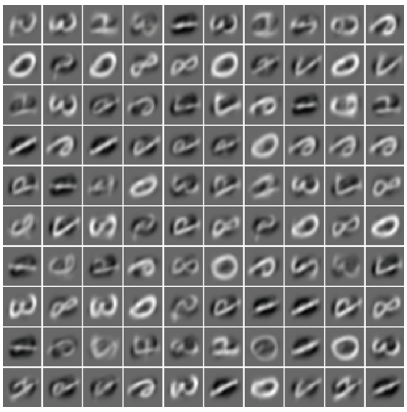

Fig. 4. The sparse features learned from reproducing kernel Hilbert space, here only 10×10 blocks extracted from input weights are demonstrated.

HP z820 workstation, Intel(R) 2×2.3G CPU E5-2630, 128G DDR3 RAM, Windows Server 2008 R2 Enterpirse, Matlab R2012b and Microsoft Visual Studio 2010.

In the experiments, for fair comparison, the structure of hidden nodes of ML-ELM and proposed method are kept the same ($[784 - 700 - 700 - 12000 - 10]$), and the DBN results were obtained using the source code in [12].

In the proposed work, the original input data is first scattered into the random reproducing kernel Hilbert space. Fig.4 shows the learned sparse features of R-ELM, and one can see that they are with sematic meanings, which is helpful for many vision tasks, such as feature extraction, object recognition, image classification, etc.

On the other hand, the learning accuracy and speed comparison of different methods are presented in Table.1.

Table 1. Performance Comparison of Different Methods

Method	Accuracy $(\%)$	Training time (s)
Proposed	**99.12**	**281.37**
ML-ELM [6]	99.01	485.87
DBN [5]	98.89	64832.13
ELM [1]	97.39	996.74

It can be seen that the original ELM has the lowest performance with only one single hidden layer, however, the training speed is pretty fast due to the random parameter settings. DBN achieves an obvious better performance than the ELM, and the reason may be the use of deep learning architecture. As for the ML-ELM, it is hundreds times faster than that of DBN. The proposed training algorithm obtained further improvement in training time (30-50%) with a better learning accuracy, and the result benefits from the training model of R-ELM and the whole hierarchical framework.

5. CONCLUSIONS

This paper proposes a fast learning algorithm for multilayer ELM, which uses R-ELM as the building block of unsupervised learning followed by an ELM-based supervised learning. The proposed R-ELM scatters the input data with a random initialization matrix, and captures the hidden features by recovering the data back from the corrupted output. To obtain more sparse hidden layer weights, ℓ_1 optimization is adopted. Compared with the existing ML-ELM and DL algorithms, our work achieves better and faster performance.

6. REFERENCES

[1] G.-B. Huang, H. Zhou, and X. Ding, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 42, pp. 513–529, 2012.

[2] A.A. Mohammed, R. Minhas, Q.M. Wu, and M.A. Ahmed, "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, pp. 2588 – 2597, 2011.

[3] C. Pan, D. Park, Y. Yang, and H. Yoo, "Leukocyte image segmentation by visual attention and extreme learning machine," *Neural Comput. and Appl.*, vol. 21, pp. 1217–1227, 2012.

[4] R. Minhas, A. Baradarani, S. Seifzadeh, and Q.M. Jonathan Wu, "Human action recognition using extreme learning machine based on visual vocabularies," *Neurocomputing*, vol. 73, pp. 1906–1917, 2010.

[5] G.E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, pp. 1527–1554, 2006.

[6] L. Kasun, H. Zhou, G.-B. Huang, and C. Vong, "Representational learning with extreme learning machine for big data," *IEEE Intelligent System*, vol. 1, pp. 160–162, 2013.

[7] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 9999, pp. 3371–3408, 2010.

[8] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Networks*, vol. 17, pp. 879–892, 2006.

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

[10] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, pp. 183–202, 2009.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[12] G.E. Hinton and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.