A Conjugate Gradient-Based Efficient Algorithm for Training Single-Hidden-Layer Neural Networks

Xiaoling Gong[1,2,3,4](✉), Jian Wang[1,2,3,4], Yanjiang Wang[1,2,3,4], and Jacek M. Zurada[1,2,3,4]

[1] College of Information & Control Engineering, China University of Petroleum, Qingdao 266580, China
s14050610@s.upc.edu.cn
[2] College of Science, China University of Petroleum, Qingdao 266580, China
[3] Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292, USA
{wangjiannl,yjwang}@upc.edu.cn
[4] Information Technology Institute, University of Social Sciences, 90-113 Łódź, Poland
jacek.zurada@louisville.edu

Abstract. A single hidden layer neural networks (SHLNNs) learning algorithm has been proposed which is called Extreme Learning Machine (ELM). It shows extremely faster than typical back propagation (BP) neural networks based on gradient descent method. However, it requires many more hidden neurons than BP neural networks to achieve assortive classification accuracy. This then leads more test time which plays an important role in practice. A novel learning algorithm (USA) for SHLNNs has been presented which updates the weights by using gradient method in the ELM framework. In this paper, we employ the conjugate gradient method to train the SHLNNs on the MNIST digit recognition problem. The simulated experiment demonstrates the better generalization and less required hidden neurons than the common ELM and USA.

Keywords: Neural network · Extreme learning machine · Conjugate gradient · MNIST

1 Introduction

Back propagation (BP) neural networks have been widely used in pattern recognition, computational intelligence. Gradient descent method is one of the most

J. Wang—This work was supported in part by the National Natural Science Foundation of China (No. 61305075), the China Postdoctoral Science Foundation (No. 2012M520624), the Natural Science Foundation of Shandong Province (No. ZR2013FQ004, ZR2013DM015), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20130133120014) and the Fundamental Research Funds for the Central Universities (No. 13CX05016A, 14CX05042A, 15CX05053A, 15CX08011A).

popular optimal techniques to train BP neural networks. Two main drawbacks such as slow convergence speed and easy trapped in local minimum constrain the BP neural networks efficiently used in some real applications.

An extremely fast learning algorithm for SHLNNs has been proposed in [1] which is named extreme learning machine (ELM). For training process, it first randomly choose the weights between input and hidden layers, then uses Moore-Penrose generalized inverse formula [2,3] to compute the output weights. The singular value decomposition (SVD) is the popular method to compute the Moore-Penrose generalized inverse. Instead of using SVD method, a conjugate gradient method was employed to compute the generalized inverse in [4]. The simulations demonstrated the faster training speed than the common ELM under the condition of the same generalization accuracy.

It is clear to see that there is no iteration step in the whole training process which is significantly different from the common BP neural networks and the support vector machine (SVM). For more details of the comparison between ELM and SVM, we refer to the survey work [5].

As a fact, ELM displays a very faster training speed and better generalization. Unfortunately, one of the main shortcomings of ELM is that the trained network model requires more hidden neurons to achieve the matched performance. An improved algorithm based on ELM, upper-layer-solution-aware (USA) algorithm, has been proposed in [9] which effectively reduces the redundant hidden neurons. The main idea of this algorithm is that it uses the gradient descent method to iteratively update the weights between in put and hidden layers. This is similar to the standard training procedure of BP neural network. The significant difference exists in the expression of the output weights. They are evaluated by using Moore-Penrose generalized inverse, which is analogous to the ELM training. It sounds like a combination of BP algorithm and ELM. Compared with BP algorithm, it enjoys a considerable faster training speed. It requires much less hidden neurons than the typical ELM. This then results in less testing time which plays an important role in real applications.

Inspired by the novelty in Yu's [9], an efficient learning algorithm based on conjugate gradient method has been presented in this paper. Gradient descent method is a special case of conjugate gradient method which engages a faster rate of convergence. Although there is a heavier computational burden of conjugate gradient method, it is restricted in a acceptable range. We compare our algorithm with ELM and USA in the MNIST digit recognition database. The simulation demonstrates that the proposed algorithm, in this paper observes significantly better generalization.

The rest of the paper is organized as follows. In Sect. 2, we give a briefly review of ELM and USA. In Sect. 3, we describe our algorithm which stems from the idea in training the SHLNNs with conjugate gradient method. The experimental results on MNIST database have been demonstrated in Sect. 4. Finally, we conclude the paper in Sect. 5.

2 Related Works

In the community of artificial neural networks, single-hidden-layer neural networks (SHLNNs) have been widely studied which include the popular networks such as radial basis function (RBF) neural networks and multi-layer perceptron (MLP) neural networks.

Given N arbitrary distinct training samples $(\mathbf{x}_j, \mathbf{t}_j)$, where $\mathbf{x}_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]^T \in \mathbf{R}^n$ are input vectors, and $\mathbf{t}_j = [t_{j1}, t_{j2}, \cdots, t_{jm}]^T \in \mathbf{R}^m$ are the desired vectors. Typically, the sigmoid function $g(x)$ is used to be the activation function of the hidden layer, while linear function $f(x) = x$ is the activation function of output layer for SHLNNs. Assume the number of hidden nodes is set to $\widetilde{N}$, then the output of a standard SHLNNs can be evaluated as follows:

$$\mathbf{y}_j = \sum_{i=1}^{\widetilde{N}} \mathbf{u}_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i), \quad j = 1, \cdots, N, \tag{1}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector from the input nodes to the ith hidden node, $\mathbf{u}_i = [u_{i1}, u_{i2}, ..., u_{in}]^T$ is the weight vector from the ith hidden node to the output nodes, and b_i is the bias of the ith hidden node.

It has beenha proved that SHLNNs with a nonpolynomial activation function can approximate (in measure) any continuous functions [10,11], which can be mathematically modeled as

$$\mathbf{y}_j = \sum_{i=1}^{\widetilde{N}} \mathbf{u}_i g_i(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j, \quad j = 1, \cdots, N, \tag{2}$$

In real applications, the neural networks are trained to get the specific $\widehat{\mathbf{w}}_i$, $\widehat{b}_i$ and $\widehat{\mathbf{u}}_i$ such that

$$\min_{\mathbf{w}_i, b_i, \mathbf{u}_i} \sum_{j=1}^{N} \|\mathbf{y}_j - \mathbf{t}_j\|^2. \tag{3}$$

There are numerous methods to implement network models which minimize the above objective function. We first introduce two efficient learning algorithm, and then present a novel one in the next section.

2.1 Extreme Learning Machine

There are increasing interests in SHLNNs with the above least square error function. A popular learning technique, extreme learning machine (ELM), was proposed in [1] which has the advantages both on simple structure and fast training speed.

Consider the total training samples together, (2) can integrated as the following matrix equation

$$\mathbf{HU} = \mathbf{T}, \tag{4}$$

where $\mathbf{H}(\mathbf{w}_i, \mathbf{b}_i)$ is the hidden layer output matrix of the neural network [12,13], $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{\widetilde{N}}]^T$ is the hidden-output layer weights matrix combined $\widetilde{N}$ vectors $\mathbf{u}_i$, and $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_N]^T$ is output matrix combined N vectors $\mathbf{t}_j$.

The hidden layer output matrix $\mathbf{H}$ can be determined by randomly chosen input-hidden layer weights $\mathbf{w}_i$ and the biases b_i. Thus, (4) is simply equivalent to a linear system. If the number of hidden layer nodes $\widetilde{N}$ is not greater than the number of training samples N, matrix $\mathbf{H}$ is invertible with probability 1. (4) has unique solution

$$\mathbf{U} = \mathbf{H}^{-1}\mathbf{T}, \tag{5}$$

which means the SHLNNs can approximate these training samples with zero error [12,13].

However, in practice, to find the minimum solution, ELM model employs the least-square method on this linear system

$$\mathbf{U} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{6}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$ [2,3],

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T. \tag{7}$$

For ELM training, the initial weights between input and hidden layers are randomly chosen instead of iteratively training, the output weights are directly computed by (6). This is significantly different from many other gradient-based learning algorithms. As a result, ELM performs much faster speed than the conventional BP training algorithms.

As a cost, ELM often requires more hidden neurons than those conventional neural networks when it achieves the matched accuracy. Sequentially, this leads to a more time-consuming situation in test procedure. It is hard to be widely used in practice.

2.2 Upper-Layer-Solution-Aware Algorithm

Upper-layer-solution-aware algorithm has been first proposed in [9] which can effectively reach good classification accuracy with a small sized architecture. The essential idea of this algorithm is that the output weights are considered as a function of input weights. The objective function can then be minimized by using gradient descent method.

For convenient, the total error function can be rewritten as follows

$$E = \|\mathbf{Y} - \mathbf{T}\|^2 = \mathrm{Tr}[(\mathbf{Y} - \mathbf{T})(\mathbf{Y} - \mathbf{T})^T]. \tag{8}$$

Note that the bias terms are implicitly represented as weight terms in which the input dimensions are augmented with 1, $\mathbf{H}$ can be simplified as

$$\mathbf{H} = g(\mathbf{W}^T\mathbf{X}), \tag{9}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_{\widetilde{N}}]$ is the input-hidden weights matrix with a combination of $\widetilde{N}$ vectors $\mathbf{w}_i$.

Similar to ELM model, the weights $\mathbf{U}$ between hidden and output layers can be expressed as follows,

$$\mathbf{U} = \mathbf{H}^{\dagger}\mathbf{T} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}. \tag{10}$$

According to (9) and (10), it is easy to see that $\mathbf{U}$ is a function of the weights $\mathbf{W}$, that is, it can be determined by the inner weights $\mathbf{W}$.

For USA algorithm, the typical gradient method was employed to find the optimal weights $\mathbf{U}$. The gradient of error function E with respect $\mathbf{W}$ is calculated as

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{W}} &= \frac{\partial \mathrm{Tr}[(\mathbf{U}^T\mathbf{H} - \mathbf{T})(\mathbf{U}^T\mathbf{H} - \mathbf{T})^T]}{\partial \mathbf{W}} \\ &= 2\mathbf{X}[\mathbf{H}^T \circ (1 - \mathbf{H})^T \circ [\mathbf{H}^{\dagger}(\mathbf{H}\mathbf{T}^T)(\mathbf{T}\mathbf{H}^{\dagger} - \mathbf{T}^T(\mathbf{T}\mathbf{H}^{\dagger}))]],\end{aligned} \tag{11}$$

where $\circ$ represents the Hadamard product.

The weight updating sequence of $\mathbf{W}$ is given by

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta\frac{\partial E}{\partial \mathbf{W}}, \quad k = 0, 1, 2, \cdots, \tag{12}$$

where η is the learning rate.

3 Conjugate Gradient-Based Efficient Algorithm

Motivated by the USA algorithm, we use the conjugate gradient method to updata the weights. In comparison with gradient descent method, conjugate gradient method has a faster convergence speed through choosing conjugate decrease direction instead of gradient direction. More importantly, it does not require computing the second order Hessian matrix which is necessary for Newton method.

For any given input weights $\mathbf{W}$, the output of the network is

$$\mathbf{Y} = \mathbf{U}^T\mathbf{H}. \tag{13}$$

The error function is defined as below

$$E = \|\mathbf{Y} - \mathbf{T}\|^2 = \mathrm{Tr}[(\mathbf{Y} - \mathbf{T})(\mathbf{Y} - \mathbf{T})^T]. \tag{14}$$

Similar to USA algorithm, the gradient of E with respect to W can be written

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{W}} &= \frac{\partial \mathrm{Tr}[(\mathbf{U}^T\mathbf{H} - \mathbf{T})(\mathbf{U}^T\mathbf{H} - \mathbf{T})^T]}{\partial \mathbf{W}} \\ &= 2\mathbf{X}[\mathbf{H}^T \circ (1 - \mathbf{H})^T \circ [\mathbf{H}^{\dagger}(\mathbf{H}\mathbf{T}^T)(\mathbf{T}\mathbf{H}^{\dagger} - \mathbf{T}^T(\mathbf{T}\mathbf{H}^{\dagger}))]].\end{aligned} \tag{15}$$

In this paper, a specific conjugate gradient method, Fletcher-Reeves [14], is used to determine the decreasing direction.

$$\mathbf{d}^k = \begin{cases} -\dfrac{\partial E(\mathbf{W}_k)}{\partial \mathbf{W}}, & \text{if} \quad k = 0, \\ -\dfrac{\partial E(\mathbf{W}_k)}{\partial \mathbf{W}} + \beta_k \mathbf{d}^{k-1}, & \text{if} \quad k \geq 1, \end{cases} \tag{16}$$

where $\frac{\partial E(\mathbf{W}_k)}{\partial \mathbf{W}}$ is the gradient of the k-th iteration, β_k is the conjugate coefficient,

$$\beta_k = \frac{\left(\frac{\partial E(\mathbf{W}_k)}{\partial \mathbf{W}}\right)^T \frac{\partial E(\mathbf{W}_k)}{\partial \mathbf{W}}}{\left(\frac{\partial E(\mathbf{W}_{k-1})}{\partial \mathbf{W}}\right)^T \frac{\partial E(\mathbf{W}_{k-1})}{\partial \mathbf{W}}} \tag{17}$$

Then the updating weight sequence of $\mathbf{W}$ is with

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta \mathbf{d}^k \tag{18}$$

where η is a constant learning rate.

This training procedure is summarized as follows

```
**Procedure of CGE Algorithm**
Require: $\mathbf{X} \in \mathbb{R}^{n\times N}$ and $\mathbf{T} \in \mathbb{R}^{m\times N}$, $g(x)$, $\widetilde{N}$, $K$
1:  $\mathbf{W}_0 \leftarrow rand(n, \widetilde{N})$
2:  for $k = 0; k < K; k++$
3:      $\mathbf{H}_k = g(\mathbf{W}_k^T \mathbf{X})$
4:      $\frac{\partial E_k}{\partial \mathbf{W}_k} = 2\mathbf{X}[\mathbf{H}_k^T \circ (1 - \mathbf{H}_k)^T \circ [\mathbf{H}_k^\dagger(\mathbf{H}_k\mathbf{T}^T)(\mathbf{T}\mathbf{H}_k^\dagger - \mathbf{T}^T(\mathbf{T}\mathbf{H}_k^\dagger))]]$
5:      $\mathbf{d}^k = \begin{cases} -\dfrac{\partial E_k}{\partial \mathbf{W}_k}, & \text{if} \quad k = 0 \\ -\dfrac{\partial E_k}{\partial \mathbf{W}_k} + \beta_k^{FR}\mathbf{d}^{k-1}, & \text{if} \quad k \geq 1 \end{cases}$
6:      $\mathbf{W}_{k+1} = \mathbf{W}_k + \eta \mathbf{d}^k$
7:  end for
8:  $\mathbf{U}_K = (\mathbf{H}_{K-1}^T)^\dagger \mathbf{T}^T$
```

4 Simulation

To verify the good performance of the proposed CGE algorithm, simulations have been done on the MNIST handwritten digital database. The results have been compared with those of ELM and USA algorithm.

4.1 Database Description

The MNIST handwritten digital database is collected from the NIST database of the National Institute of Standards and Technology, and its each digital image has been normalized to an image for 28×28. The data set is stored in the form of matrix, in which the each sample, i.e., each handwritten number, is a 1×784 vector. Each element in the vector is a number between 0~255, representing the gray levels of each pixel. The MNIST database has a total number of 60000 for training samples and 10000 for testing samples.

Table 1. Accuracy (%) comparison for different algorithms

Algorithms	Hidden neurons	Training accuracy	Testing accuracy
ELM	64	67.13	67.88
ELM	128	78.32	78.99
ELM	256	85.08	85.55
ELM	512	89.50	89.65
ELM	1024	92.74	92.65
ELM	2048	95.32	94.68
USA	64	85.80	84.27
USA	128	89.71	88.06
USA	256	92.97	90.82
USA	512	95.18	93.79
USA	1024	97.62	95.82
USA	2048	98.93	97.86
CGE	64	86.70	85.89
CGE	128	90.40	89.62
CGE	256	93.35	92.68
CGE	512	96.03	95.58
CGE	1024	97.87	97.63
CGE	2048	98.99	98.95

4.2 Experimental Results

We compared the ELM, USA and the proposed CGE algorithms with same number of hidden neurons: 64, 128, 256, 512, 1024 and 2048. Each network configuration was run 10 times to calculate the mean values of classification accuracies both on training and test sets. Starting with the same initial weights $\mathbf{W}$ for these three different algorithms, ELM evaluated the output weights $\mathbf{U}$ at once and with non-iteration. For USA and CGE, the identical stop criteria are configured to guarantee fairly comparison.

The results summarized in Table 1 are compared from two aspects, training and testing accuracies, based on same hidden neurons. It is clear to see that the accuracies are monotonically grown with the increasing number of hidden neurons. We observe that CGE and USA perform much better than the typical ELM mode. And CGE is slightly better than USA algorithm.

To make a simple observation, the testing accuracies have been graphed in Fig. 1 which intuitively describe the generalization abilities of these three algorithms. We can obtain different interesting conclusions from the following two perspectives. (1) If we draw vertical lines in Fig. 1, we can observe th different accuracies with same number of hidden neurons. Obviously, the proposed CGE algorithm performs much better than USA and ELM. (2) If we draw horizontal

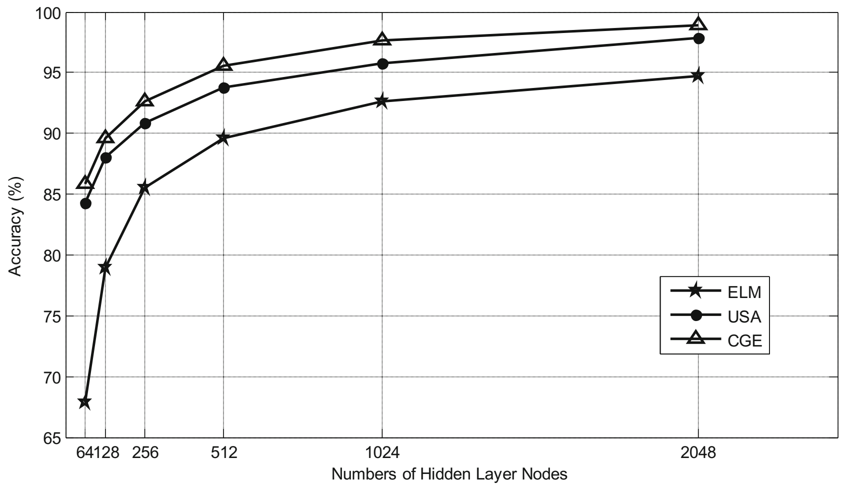

Fig. 1. The testing accuracy of different learning algorithms on the MNIST database.

lines in Fig. 1, we can get that different number of hidden neurons are required to reach the matched accuracy for these three algorithm. It apparently indicates that CGE algorithm may attain the simplest network architecture among these algorithms.

5 Conclusions

In this paper, a novel efficient learning algorithm, CGE, has been proposed for SHLNNs which is motivated by the USA algorithm. Instead of using gradient descent method, a specific conjugate gradient method, F-R, has been employed to train the networks. The popular digital dataset, MNIST, has been used to verify the advantages of CGE. The simulations demonstrate that CGE performs much better than its counterparts, ELM and USA, and results in simplest network.

References

1. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(123), 489–501 (2006)
2. Serre, D.: Matrices: Theory and Applications. Springer, New York (2002)
3. Rao, C.R., Mitra, S.K.: Generalized Inverse of Matrices and Its Applications. Wiley, New York (1971)
4. Zhang, P., Wang, X., Gu, D., Zhao, S.: Extreme learning machine based on conjugate gradient. J. Comput. Appl. **35**(10), 2757–2760 (2015)
5. Chorowski, J., Wang, J., Zurada, J.M.: Review and performance comparison of SVM- and ELM-based classifiers. Neurocomputing **128**(5), 507–516 (2014)
6. Bartlett, P.L.: The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans. Inf. Theor. **44**(2), 525–536 (1998)

7. Widrow, B., Greenblatt, A., Kim, Y., Park, D.: The no-prop algorithm: a new learning algorithm for multilayer neural networks. Neural Netw. **37**, 182–188 (2013)
8. Zhu, Q.Y., Qin, A.K., Suganthan, P.N., Huang, G.B.: Evolutionary extreme learning machine. Pattern Recogn. **38**(10), 1759–1763 (2005)
9. Yu, D., Deng, L.: Efficient and effective algorithms for training single-hidden-layer neural networks. Pattern Recogn. Lett. **33**(5), 554–558 (2012)
10. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991)
11. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. **6**(6), 861–867 (1993)
12. Huang, G.B., Babri, H.A.: Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans. Neural Netw. **9**(1), 224–229 (1998)
13. Huang, G.B.: Learning capability and storage capacity of two hidden-layer feedforward networks. IEEE Trans. Neural Netw. **14**(2), 274–281 (2003)
14. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**, 149–154 (1964)