Neurocomputing 72 (2009) 3066–3076

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Partial Lanczos extreme learning machine for single-output regression problems

Xiaoliang Tang, Min Han *

*School of Electronic and Information Engineering, Dalian University of Technology, 2 Linggong Lu, Ganjingzi Qu, Dalian 116023, China*

ARTICLE INFO

*Article history:*
Received 6 January 2009
Received in revised form
27 March 2009
Accepted 28 March 2009
Communicated by: G.-B. Huang
Available online 17 April 2009

*Keywords:*
Extreme learning machine
Lanczos bidiagonalization
Singular value decomposition
Regularization
Generalized cross validation

ABSTRACT

There are two problems preventing the further development of extreme learning machine (ELM). First, the ill-conditioning of hidden layer output matrix reduces the stability of ELM. Second, the complexity of singular value decomposition (SVD) for computing Moore–Penrose generalized inverse limits the learning speed of ELM. For these two problems, this paper proposes the partial Lanczos ELM (PL-ELM) which employs the hybrid of partial Lanczos bidiagonalization and SVD to compute output weights. Experimental results indicate that, compared with ELM, PL-ELM not only effectively improves the stability and generalization performance but also raises the learning speed.



## 1. Introduction

Recently, Huang et al. [1,2] have proposed a novel learning algorithm for single hidden layer feedforward networks (SLFNs) named extreme learning machine (ELM). In ELM, input weights and hidden layer bias are randomly chosen, and output weights are analytically determined based on the Moore–Penrose generalized inverse of hidden layer output matrix. ELM provides better generalization performance with extremely fast learning speed than gradient-based learning algorithms. In addition, ELM avoids many difficulties faced by gradient-based learning methods such as stopping criteria, learning rate, learning epochs and local minima [1,3]. Depending upon these advantages, ELM has been successfully applied in many areas, such as classification [4], function approximation [5,6], nontechnical loss analysis [7], terrain reconstruction [8] and protein structure prediction [9].

In order to enhance the performance of ELM, many improved models of ELM have been proposed. We roughly summarize these improved models of ELM into four types: incremental type, optimization type, replacement type and ensemble type. The incremental type of ELM creates new hidden layer neurons one by one according to certain criteria, which is very suitable for stream data [10–12]. The optimization type of ELM employs some techniques, such as evolutionary algorithm and linear programming, to tune input weights and hidden layer bias and optimize the network structure; it achieves good generalization performance and much more compact network structure [13–15]. The replacement type of ELM replaces the activation functions of ELM (sigmoid and RBF) with sine and cosine functions (or other functions), which is helpful to improve the accuracy and the convergence rate for the problem of function approximation [6]. In the ensemble type of ELM, different ELMs are trained by disjoint subsets of data, but they can share the same hidden layer neurons [5,16].

Although these models improve the performance of ELM to a certain degree, there still exists great development space for the stability and the learning speed of ELM. We firstly analyze the critical factor influencing the stability of ELM.

As rigorously proven by Huang et al. [1], input weights and hidden layer bias of SLFNs can be randomly chosen without tuning, and then the hidden layer output matrix H of SLFNs can remain unchanged in the whole learning process. Thus, training a SLFN is equivalent to finding the minimal Euclidean norm solution $\beta$ of the linear system $H\beta = T$ [1]:

$$\beta = H^{\dagger}T \quad \text{such that } \beta = \arg\min_{\beta} \|H\beta - T\|_2 \tag{1}$$

where $\beta$ denotes the output weight matrix of the trained ELM, $H^{\dagger}$ denotes the Moore–Penrose generalized inverse of the hidden layer output matrix H, T denotes the target vector. The detailed specification of these variables is given in [1].

* Corresponding author. Tel.: +86 411 84708719; fax: +86 411 84707847.
*E-mail address:* minhan@dlut.edu.cn (M. Han).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.03.016

ELM computes the Moore–Penrose generalized inverse $H^\dagger$ in Eq. (1) based on the singular value decomposition (SVD) of H [17]. The SVD of $H \in \mathbb{R}^{N\times n}$, $N \geqslant n$, is given by

$$H = U\Sigma V^T = \sum_{i=1}^{n} u_i \sigma_i v_i^T \tag{2}$$

where $U = (u_1, \ldots, u_n)$, $V = (v_1, \ldots, v_n)$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, $\sigma_i \in \Sigma$, $i = 1, \ldots, n$ are singular values of H with the order $\sigma_1 \geqslant \cdots \geqslant \sigma_n \geqslant 0$, and the rank $r$ of H equals the number of strictly positive singular values, i.e. $\sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$.

Based on Eq. (2), the Moore–Penrose generalized inverse $H^\dagger$ in Eq. (1) is computed by

$$H^\dagger = V\Sigma^\dagger U^T = \sum_{i=1}^{r} \frac{v_i u_i^T}{\sigma_i} \tag{3}$$

and the output weight matrix of ELM is computed by

$$\beta = H^\dagger T = \sum_{i=1}^{r} \frac{v_i u_i^T}{\sigma_i} T \tag{4}$$

In practical applications, the target vector T usually contains a certain degree of perturbations. Let $e$ denote the perturbation in T and $\tilde{T} = T + e$ denote the perturbed target vector. The perturbed output weight matrix $\tilde{\beta}$ of ELM can be computed by

$$\tilde{\beta} = H^\dagger \tilde{T} = H^\dagger T + H^\dagger e = \sum_{i=1}^{r} \frac{v_i u_i^T}{\sigma_i} T + \sum_{i=1}^{r} \frac{v_i u_i^T}{\sigma_i} e \tag{5}$$

When the hidden layer output matrix H has a very large condition number (i.e. H tending to be ill-conditioned), there always exist some very small (positive and approximate to zero) singular values in H. From the two summation items in the right hand of Eq. (5) we can find that, when divided by small singular values, the output weights become very large and tend to be greatly influenced by the perturbation $e$. The large output weights also weaken the generalization capability of ELM, because the trained network will behave very different if test data change but slightly away from the training data. The ill-conditioning of hidden layer output matrix is the critical factor influencing the stability of ELM.

We then analyze the critical factor limiting the learning speed of ELM as follows. The learning time of ELM is mainly consumed by computing the Moore–Penrose generalized inverse $H^\dagger$ [1]. As mentioned above (see Eqs. (2)–(4)), computing $H^\dagger$ requires the SVD of H. For matrix $H \in \mathbb{R}^{N\times n}$, the computational complexity of SVD is $O(4Nn^2 + 8n^3)$ [18]. When the number of hidden layer neurons (i.e. $n$) becomes large, the computational complexity of SVD will significantly rise. Although some methods are proposed to compact the network structure of ELM [13,14], they make ELM lose its property of determining network structure completely independent from training data. The large computational complexity of SVD is the critical factor limiting the learning speed of ELM.

To improve the stability of ELM, it is necessary to incorporate some regularization methods for reducing the influences of ill-conditioning and perturbations. Truncated SVD (TSVD) and Tikhonov regularization [19,20] are two kinds of widely used regularization methods. As proven by [19,21], TSVD can provide similar results with Tikhonov regularization. TSVD only focuses on the contributions associated with the $k$ largest singular values and effectively avoids the adverse effects caused by the smallest singular values. According to TSVD, the Moore–Penrose generalized inverse of H can be computed by

$$H^\dagger = \sum_{i=1}^{\kappa} \frac{v_i u_i^T}{\sigma_i}, \quad \kappa \leqslant r = \mathrm{rank}(H) \leqslant n \tag{6}$$

where $\kappa$ is the truncation number of TSVD for H, usually $\kappa \ll n$.

To improve the learning speed of ELM, it is necessary to reduce the computational complexity of SVD. Although TSVD provides more stable solutions than SVD, it still depends on the results of SVD and has the similar computational complexity with SVD [22]. Lanczos bidiagonalization (LBD) is an efficient iterative method for computing the SVD of large and ill-conditioned matrices [23–25]. Given the appropriate number of iterations, we can conduct partial LBD for a matrix. Partial LBD not only makes the computation of SVD fairly inexpensive but also provides good approximation to the singular triplets associated with the largest singular values of the matrix [26]. Partial LBD has been applied to the computation of many regularization methods, such as Tikhonov regularization and TSVD [27–29].

In this paper, we propose an enhanced ELM, called partial Lanczos ELM (PL-ELM), which computes the output weights based on the hybrid of partial LBD and SVD. PL-ELM first implements partial LBD to the hidden layer output matrix H so that the linear system $H\beta = T$ can be projected onto a small-size Krylov subspace, then PL-ELM determines the approximate solution of output weights from the Krylov subspace. Since the dimension of the Krylov subspace is usually much smaller than that of hidden layer output matrix, PL-ELM can significantly reduce the computational complexity compared with ELM. On the other hand, the results of PL-ELM are very similar to those achieved by directly applying TSVD to the linear system $H\beta = T$. Currently, PL-ELM mainly deals with the single-output regression problem. We validate the performance of PL-ELM using several benchmark regression data sets.

This paper is organized as follows. Section 2 briefly reviews the partial LBD algorithm; Section 3 proposes the PL-ELM algorithm and then analyzes the computational complexity, the relative perturbation bound and the parameter choice of the proposed PL-ELM algorithm; Section 4 shows experimental results and discussions; Section 5 contains the conclusion and the consideration of future research.

## 2. Review of partial LBD

This section briefly reviews partial LBD. The detailed descriptions of partial LBD can be found in [23,30,31] with slightly different notations. The present notation in this section is consistent with that of ELM [1]. Given the matrix $H \in \mathbb{R}^{N\times n}$ and the iteration number $k (k < n)$, LBD generates a sequence of Lanczos vectors $u_j \in \mathbb{R}^N$ and $v_j \in \mathbb{R}^n$ and scalars $\alpha_j$ and $\gamma_j$, $(j = 1, \ldots, k)$:

Choose a nonzero starting vector $p_0 \in \mathbb{R}^N$, let $\gamma_1 = \|p_0\|_2$, $u_1 = p_0/\gamma_1$ and $v_0 \equiv 0$. Then implement Lanczos iterations for $j = 1, \ldots, k$

$$\alpha_j v_j = H^T u_j - \gamma_j v_{j-1} \tag{7}$$

$$\gamma_{j+1} u_{j+1} = H v_j - \alpha_j u_j \tag{8}$$

After $k$ Lanczos iterations, the lower bidiagonal matrix $B_k$ and two orthonormal Lanczos matrices $U_{k+1}$ and $V_k$ are generated by

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \gamma_2 & \alpha_2 & & \\ & \gamma_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \gamma_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1)\times k} \tag{9}$$

$$U_{k+1} = (u_1, u_2, \ldots, u_{k+1}) \in \mathbb{R}^{N\times(k+1)},$$
$$U_{k+1}^T U_{k+1} = I_{k+1}, \quad u_1 = p_0/\gamma_1 \tag{10}$$