Neurocomputing 151 (2015) 933–942

Contents lists available at ScienceDirect

Neurocomputing

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

An effective semi-cross-validation model selection method for extreme learning machine with ridge regression

Zhifei Shao [a], Meng Joo Er [a,*], Ning Wang [b]

[a] School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore
[b] Marine Engineering College, Dalian Maritime University, Dalian 116026, China

ARTICLE INFO

Article history:
Received 4 October 2013
Received in revised form
28 September 2014
Accepted 2 October 2014
Available online 14 October 2014

Keywords:
Extreme learning machine
Ridge regression
Regularized ELM

ABSTRACT

Extreme Learning Machine (ELM) has attracted comprehensive attentions as a universal function approximator with its extremely fast learning speed and good generalization performance. Compared to other learning methods for Single Layer Feedforward Networks (SLFNs), the unique feature of the ELM is that the input parameters of hidden neurons are randomly generated rather than being iteratively tuned, and thereby dramatically reducing the computational burden. However, it has been pointed out that the randomness of the ELM parameters would result in fluctuating performance. In this paper, we systematically investigate the performance stabilization effect brought by a regularized variant of the ELM, named Regularized ELM (RELM). Furthermore, by using the PREdiction Sum of Squares (PRESS) statistics formula and a unique property of the RELM, we propose a semi-cross-validation algorithm to effectively realize a robust RELM-based model selection for SLFNs, termed as Automatic Regularized Extreme Learning Machine with Leave-One-Out cross-validation (AR-ELM-LOO). The simulation results show that the AR-ELM-LOO can significantly reduce the randomness performance of the ELM and it can produce nearly identical results as the full cross-validation procedure.

1. Introduction

Extreme Learning Machine (ELM) achieves its extremely fast learning speed through random generation of the input parameters of hidden neurons, and as a result, the ELM has attracted a great deal of research attention in recent years. However, one of the biggest concerns towards the ELM is also the reason of its popularity: randomness nature. Indeed, a non-analytical determination of some very important parameters can make the algorithm appear less secure. Zhu et al. pointed out that the random assignment of the ELM parameters can introduce non-optimal solutions [36]. Fluctuating performance of the ELM was also reported with different initial parameters [16,28,35]. Especially for sparse data, the effect of random parameters imposed on the generalization performance can be quite significant [29].

Various approaches have been applied to reduce the random effect of the ELM. Evolutionary algorithms, as a traditional parameter selection method, are no surprise to be used to address this problem [36,29]. However, their slow learning speed becomes the performance bottleneck and eliminates the advantage of the ELM against other learning methods. Another approach is to use an ensemble of ELMs, which gives an output by combining them together [16,21,24,28]. The most common approach is to simply compute the average outputs of the ensemble [16]. Therefore, statistically speaking, the overall performance is at the same level as the original ELM, but with smaller output fluctuating region. Based on the knowledge that smaller output weights could lead to better generalization ability for ELMs with the same training performance [14], a subset of ELMs is selected to create an ensemble [21]. Similarly, Miche et al. also used a subset of generated ELMs to create the ensemble structure, but with a more complex selection procedure [24]. The third approach, resembling the common method used in constructive neural networks, is to first create a large pool of neurons, and a subset of more significant neurons is selected using various ranking algorithms [6,17,18]. A reverse approach is also popular, where an ELM structure with more neurons than necessary is first created, then pruning methods can be applied to delete the unimportant ones [24,18,33,34,26,20].

Besides relying methods outside the ELM structure to enhance its performance, a major tweak, ridge regression [11], was implemented in the ELM [12]. It has been found out that the generalization ability of Regularized ELM (RELM) is less sensitive to the choice of number of neurons L than the traditional ELM. And for some activation function, sigmoid for instance, it appears that the generalization performance reaches a plateau rather than deteriorating when L exceeds some value [13].

Compared to the ELM, two main advantages of RELM have been previously reported, i.e., the improvement of generalization

* Corresponding author.
E-mail addresses: zshao1@ntu.edu.sg (Z. Shao), emjer@ntu.edu.sg (M.J. Er), n.wang.dmu.cn@gmail.com (N. Wang).

http://dx.doi.org/10.1016/j.neucom.2014.10.002
0925-2312/

performance and the ease of parameter selection [13]. In this paper, we systematically investigate an additional benefit brought by this regularized version of the ELM, that is the great improvement on the performance stability of the ELM, provided with different initial randomly generated parameters. And thus we argue that RELM is a fair alternative to the ELM ensemble in terms of producing more consistent results.

Another important issue is to find the optimal value of ridge parameter of RELM, and the research on this field has been going on for decades. Well-known algorithms include graphical driven and data driven methods [23]. However, they cannot provide robust solutions since graphical driven methods require subjective judgements and data driven methods often fail to find the true optimal value [23]. In this paper, we implement the Leave-One-Out (LOO) cross-validation method with the help of PREdiction Sum of Squares (PRESS) statistic formula to tackle this issue. Since PRESS formula provides a very efficient way of calculating the LOO error [25], the overall computational requirement of this cross-validation approach is reduced to a level similar to the graphical driven methods, but with much more improved results.

wFurthermore, by utilizing a unique property observed in the RELM, we propose an algorithm termed as Automatic RELM with LOO (AR-ELM-LOO), which effectively selects a subset of ridge parameters from the potential candidates and further reduces 2/3 of computational power and generate almost the same results compared to the normal full cross-validation procedure. The remaining of this paper is organized as follows: the preliminaries of the ELM, the RELM, and ensemble methods are given in Section 2, and in Section 3, their fluctuating performance caused by different initial parameters is compared. The AR-ELM-LOO algorithm is introduced in Section 4. Simulation results on benchmark datasets are presented and analyzed in Section 5. Conclusions are drawn in Section 6.

2. Preliminaries

ELM is a novel algorithm for Single Layer Feedforward Networks (SLFNs) [12]. Its salient feature is that the input weights and hidden biases are randomly chosen instead of exhaustively tuned, and the output weights are analytically determined using *Moore–Penrose generalized inverse* [13]. The ELM aims to reach the smallest training error as well as the smallest norm of output weights. Consequently, it is able to provide better generalization performance with much faster learning speed and avoid traditional ANN tuning issues such as learning rate, stopping criterion, number of learning epochs and local minima [14,8,19,15]. In this section, the preliminaries of the ELM, the ELM ensemble and the RELM are introduced.

2.1. Original ELM

The structure of the original ELM is shown in Fig. 1. For the sake of simplicity, the usual setup of the ELM for regression with single output is considered in this paper.

The output y with L hidden nodes can be represented by

$$y = \sum_{i=1}^{L} \beta_i g_i(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\boldsymbol{\omega}_i, b_i, \mathbf{x}) = \mathbf{H}\boldsymbol{\beta} \tag{1}$$

where $\mathbf{x}$ is the input sample and $(\boldsymbol{\omega}_i, b_i)$ are the randomly generalized input parameters; $\mathbf{x}, \boldsymbol{\omega}_i \in \mathbb{R}^d$ and g_i denotes the output of the ith hidden node output function $G(\boldsymbol{\omega}_i, b_i, \mathbf{x})$; $\mathbf{H}$ and $\boldsymbol{\beta}$ are the hidden layer output matrix and the output weights matrix, respectively. For N distinct samples (x_j, t_j), $j = 1, \ldots N$, Eq. (1) can be written as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{2}$$

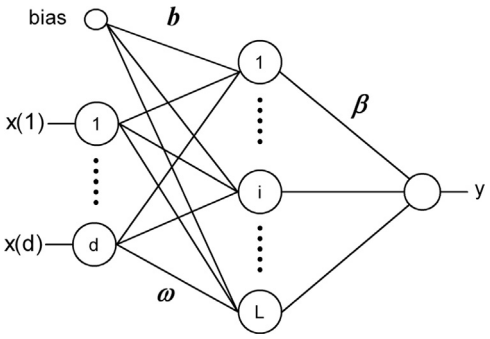

Fig. 1. The ELM network structure.

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_N \end{bmatrix} = \begin{bmatrix} \mathbf{h}(x_1) \\ \vdots \\ \mathbf{h}(x_N) \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{\omega}_1, b_1, x_1) & \cdots & G(\boldsymbol{\omega}_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{\omega}_1, b_1, x_N) & \cdots & G(\boldsymbol{\omega}_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{3}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \end{bmatrix}_{L \times 1} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}_{N \times 1} \tag{4}$$

where $\mathbf{T}$ is the target matrix.

Since the input weights of its hidden neurons $(\boldsymbol{\omega}_i, b_i)$ can be randomly generated instead of tuned [13], the only parameter that needs to be calculated in the ELM is the output weights matrix $\boldsymbol{\beta}$, which can be easily done through Least Squares Estimate (LSE):

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the *Moore–Penrose generalized inverse* of matrix $\mathbf{H}$, which can be calculated through orthogonal projection.

The procedure of the ELM goes as follows:

1. Randomly generate hidden neuron parameters $(\boldsymbol{\omega}, \boldsymbol{\beta})$.
2. Calculate hidden layer output matrix $\mathbf{H}$ through Eq. (3).
3. Calculate the output weights $\boldsymbol{\beta}$ using Eq. (5).

2.2. ELM ensemble

The idea of neural network ensembles was first introduced by Hansen and Salamon [9]. By combining the results of an ensemble of neural networks, it has been shown that the overall network performance can be expected to improve. Because of the ease of implementation and relatively low computational requirement, ensemble methods have been applied to the ELM to reduce its fluctuating performance [16,21,28]. The common scheme of ELM ensemble is shown in Fig. 2.

The ELM ensemble structure consists of P individual ELMs, where the input parameters $(\boldsymbol{\omega}^j, b^j)$, $j \in [1, P]$ are randomly generated and their output weights $\boldsymbol{\beta}^j$ are analytically determined based on the training data. Although numerous ways exist in generating the final output, the common one is the average of each individual ELM's result [16], for being the easiest and, most of the time, effective approach. Other methods include bagging [4] and boosting [27,7], and even GA [36,32]. But they may significantly reduce the learning efficiency of the ELM.