Neurocomputing 167 (2015) 671–687

Contents lists available at ScienceDirect

Neurocomputing

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

An accelerating scheme for destructive parsimonious extreme learning machine

Yong-Ping Zhao [a,*], Bing Li [a], Ye-Bo Li [b]

[a] School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[b] AVIC Aeroengine Control Research Institute, Wuxi 214063, China

ARTICLE INFO

Article history:
Received 27 December 2014
Received in revised form
10 February 2015
Accepted 2 April 2015
Communicated by G.-B. Huang
Available online 10 April 2015

Keywords:
Single-hidden layer feedforward network
Extreme learning machine
Destructive algorithm
Constructive algorithms
Sparseness

ABSTRACT

Constructive and destructive parsimonious extreme learning machines (CP-ELM and DP-ELM) were recently proposed to sparsify ELM. In comparison with CP-ELM, DP-ELM owns the advantage in the number of hidden nodes, but it loses the edge with respect to the training time. Hence, in this paper an equivalent measure is proposed to accelerate DP-ELM (ADP-ELM). As a result, ADP-ELM not only keeps the same hidden nodes as DP-ELM but also needs less training time than CP-ELM, which is especially important for the training time sensitive scenarios. The similar idea is extended to regularized ELM (RELM), yielding ADP-RELM. ADP-RELM accelerates the training process of DP-RELM further, and it works better than CP-RELM in terms of the number of hidden nodes and the training time. In addition, the computational complexity of the proposed accelerating scheme is analyzed in theory. From reported results on ten benchmark data sets, the effectiveness and usefulness of the proposed accelerating scheme in this paper is confirmed experimentally.

1. Introduction

The widespread popularity of single-hidden layer feedforward networks (SLFNs) in extensive fields is mainly due to their powerfulness of approximating complex nonlinear mappings and simple forms. As a specific type of SLFNs, extreme learning machine (ELM) [1–3] recently has drawn a lot of interests from researchers and engineers. Generally, training an ELM consists of two main stages [4]: (1) random feature mapping and (2) linear parameters solving. In the first stage, ELM randomly initializes the hidden layer to map the input samples into a so-called ELM feature space with some nonlinear mapping functions, which can be any nonlinear piecewise continuous functions, such as the sigmoid and the RBF. Since in ELM the hidden node parameters are randomly generated (independent of the training samples) without tuning according to any continuous probability distribution instead of being explicitly trained, it owns the remarkable computational priority to regular gradient-descent backpropagation [5,6]. That is, unlike conventional learning methods that must see the training samples before generating the hidden node parameters, ELM can generate the hidden node parameters before seeing the training samples. In the second stage, the linear parameters can be obtained by solving the Moore-Penrose generalized inverse of hidden layer output matrix, which reaches both the smallest training error and the smallest norm of output weights [7]. Hence, different from most algorithms proposed for feedforward neural networks not considering the generalization performance when they were proposed first time, ELM can achieve better generalization performance.

From the interpolation perspective, Huang et al. [3] indicated that for any given training set, there exists an ELM network which gives sufficient small training error in squared error sense with probability one, and the number of hidden nodes is not larger than the number of distinct training samples. If the number of hidden nodes amounts to the number of distinct training samples, then training errors decreases to zero with probability one. Actually, in the implementation of ELM, it is found that the generalization performance of ELM is not sensitive to the number of hidden nodes and good performance can be reached as long as the number of hidden nodes is large enough [8]. In addition, unlike traditional SLFNs, such as radial basis function neural networks and multilayer perceptron with one hidden layer, where the activation functions are required to be continuous or differentiable, ELM can choose the threshold function and many others as activation functions without sacrificing the universal approximation capability at all. In statistical learning theory, Vapnik–Chervonenkis (VC) dimension theory is one of the most widely used frameworks in generalization bound analysis. According to the structure risk minimization perspective, to obtain better generalization performance on testing set, an algorithm should not only

* Corresponding author.
E-mail address: y.p.zhao@163.com (Y.-P. Zhao).

http://dx.doi.org/10.1016/j.neucom.2015.04.002
0925-2312/

Table 1
The computational complexity of ADP-ELM/ADP-RELM at the ith iteration.

Algorithms	ADP-ELM/ADP-RELM at the ith iteration			
Operation	Multiplication ($\times$)	Addition ($+$)	Division (/)	Square root ($\sqrt{\ }$)
$\left(\boldsymbol{R}^{(i-1)}\right)^{-1}$	$\frac{1}{6}L_i^3-\frac{1}{6}L_i$	$\frac{1}{6}L_i^3-\frac{1}{6}L_i$	$\frac{1}{2}L_i^2+\frac{1}{2}L_i$	0
$\hat{\boldsymbol{\beta}}^{(i-1)}$	$\frac{1}{2}L_i(L_i+1)m$	$\frac{1}{2}L_i(L_i-1)m$	0	0
$\frac{\left(\hat{\beta}_{j_s}^{(i-1)}\right)^2}{\|\boldsymbol{r}_{j_s}^{*}(i-1)\|_2^2}, j_s=1,\cdots,L_i$	$\frac{1}{2}L_i^2+\frac{1}{2}L_i+L_im$	$\frac{1}{2}L_i^2-\frac{3}{2}L_i+L_im$	L_i	0
In sum	$\frac{1}{6}L_i^3+\frac{1}{2}L_i^2+\frac{1}{3}L_i+\frac{3}{2}L_im+\frac{1}{2}L_i^2m$	$\frac{1}{6}L_i^3+\frac{1}{2}L_i^2-\frac{5}{3}L_i+\frac{1}{2}L_i^2m+\frac{1}{2}L_im$	$\frac{1}{2}L_i^2+\frac{3}{2}L_i$	0

Note: $L_i=L-i+1$.

Table 2
The computational complexity of DP-ELM/DP-RELM at the ith iteration.

Algorithms	DP-ELM/DP-RELM at the ith iteration			
Operation	Multiplication ($\times$)	Addition ($+$)	Division (/)	Square root ($\sqrt{\ }$)
$\left[\boldsymbol{R}_{-s,j_s}^{(i)} \quad \boldsymbol{T}_{j_s}^{(i-1)}\right]=\boldsymbol{Q}_{j_s}^{(i)}\begin{bmatrix}\boldsymbol{R}_{j_s}^{(i)} & \boldsymbol{T}_{j_s}^{(i)}\\ \boldsymbol{0}_{1\times(L-i-j_s+1)} & \boldsymbol{t}_{i_s}^{(i)}\end{bmatrix} j_s=1,\cdots,L_i-1$	$\frac{2}{3}L_i^3+L_i^2-\frac{5}{3}L_i+2L_i^2m-2L_im$	$\frac{1}{3}L_i^3+\frac{1}{2}L_i^2-\frac{5}{6}L_i+L_i^2m-L_im$	$L_i^2-L_i$	$\frac{1}{2}L_i^2-\frac{1}{2}L_i$
$\|\boldsymbol{t}_{i_s}^{(i)}\|_F^2\ j_s=1,\cdots,L_i$	L_im	$L_i(m-1)$	0	0
In sum	$\frac{2}{3}L_i^3+L_i^2-\frac{5}{3}L_i+2L_i^2m-L_im$	$\frac{1}{3}L_i^3+\frac{1}{2}L_i^2-\frac{11}{6}L_i+L_i^2m$	$L_i^2-L_i$	$\frac{1}{2}L_i^2-\frac{1}{2}L_i$

Table 3
Specifications on each benchmark data set.

Data sets	Hidden node type	#Hidden nodes	$\log_2(\mu)$	#Training	#Testing	#Inputs	#Outputs
Energy efficiency	Sigmoid	190	−15	450	318	8	2
	RBF	220	−20	450	318	8	2
0.5							
Sml2010	Sigmoid	80	−2	3000	1137	16	2
	RBF	90	−8	3000	1137	16	2
0.5							
Parkinsons	Sigmoid	100	−10	4000	1875	16	2
	RBF	70	−13	4000	1875	16	2
0.5							
Airfoil	Sigmoid	100	−17	800	703	5	1
	RBF	90	−30	800	703	5	1
0.5							
Abalone	Sigmoid	40	−11	2800	1377	8	1
	RBF	60	−10	2800	1377	8	1
0.5							
Winequality white	Sigmoid	90	−8	3000	961	11	1
	RBF	70	−11	3000	961	11	1
0.5							
CCPP	Sigmoid	220	−34	6000	3527	4	1
	RBF	230	−35	6000	3527	4	1
0.5							
Ailerons	Sigmoid	120	−1	7154	6596	40	1
	RBF	100	−10	7154	6596	40	1
0.5							
Elevators	Sigmoid	60	−4	8752	7847	18	1
	RBF	50	−11	8752	7847	18	1
0.5							
Pole	Sigmoid	300	−6	10000	4958	26	1
	RBF	300	−27	10000	4958	26	1

Notes: #Training represents the number of training samples, #Testing represents the number of testing samples, #Inputs represents the number of input variables, and #Outputs represents the number of output variables.

achieve low training error on training set, but also should have a lower VC dimension. Aiming to classification tasks, Liu et al. [9] proved that the VC dimension of an ELM is equal to its number of hidden nodes with probability one, which states that ELM has a relatively low VC dimension. With respect to regression problems, the generalization ability of ELM had been comprehensively studied in [10] and [11], leading to the conclusion that ELM with some suitable activation functions, such as polynomials, sigmoid and Nadaraya–Watson function, can achieve the same optimal generalization bound as a SLFN in which all parameters are tunable. From the above analyses, it is known that ELM owns excellent universal approximation capability and generalization performance. However, due to the fact that ELM generates hidden nodes randomly, it usually requires more hidden nodes than traditional SLFN to get matched performance. Large network size always signifies more running time in the testing phase. In cost sensitive learning, the testing time