Neurocomputing 198 (2016) 20–26

Contents lists available at ScienceDirect

Neurocomputing

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

Preliminary study on Wilcoxon-norm-based robust extreme learning machine

Xiao-Liang Xie *, Gui-Bin Bian, Zeng-Guang Hou, Zhen-Qiu Feng, Jian-Long Hao

State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 1 April 2015
Received in revised form
3 November 2015
Accepted 6 December 2015
Available online 8 March 2016

Keywords:
Extreme learning machine
Wilcoxon neural network
Wilcoxon-norm based robust extreme learning machine

ABSTRACT

The fact that the linear estimators using the rank-based Wilcoxon approach in linear regression problems are usually insensitive to outliers is known in statistics. Outliers are the data points that differ greatly from the pattern set by the bulk of the data. Inspired by this fact, Hsieh et al. introduced the Wilcoxon approach into the area of machine learning. They investigated four new learning machines, such as Wilcoxon neural network (WNN), and developed four gradient descent based backpropagation algorithms to train these learning machines. The performances of these machines are better than ordinary nonrobust neural networks in outliers exist tasks. However, it is hard to balance the learning speed and the stability of these algorithms which is inherently the drawback of gradient descent based algorithms. In this paper, a new algorithm is used to train the output weights of single-layer feedforward neural networks (SLFN) with input weights and biases being randomly chosen. This algorithm is called Wilcoxon-norm based robust extreme learning machine or WRELM for short.

1. Introduction

It is reported that the modern age of neural network began with the work of McCulloch and Pitts in 1943 [1]. Since then, some popular and powerful artificial neural networks (ANN) have been proposed, such as self organizing maps (SOM) [2], radial basis function neural networks (RBF) [3], and support vector machines (SVM) [4]. Several learning algorithms have been proposed in the literature for training the aforementioned learning machines [2–6]. Among these machines, one simple structure is multilayer perceptron artificial neural networks (MLP). Some off-line algorithms have been introduced to learn the weights and biases of MLP. One well-known gradient descent based batch learning algorithm is back-propagation (BP) [5]. In order to improve the convergence speed of BP algorithm, several improvements were made in [6,7]. One problem associated with MLP is how to decide the stop criterion of training process, and another problem is how to decide the number of hidden layers and the number of neurons in each layer. It has been proved that a single-hidden layer feedforward neural network with additive hidden nodes and with a nonpolynomial activation function can approximate any continuous function in a compact set [8]. Huang et al. rigorously proved that SLFNs with randomly assigned input weights and hidden neurons' biases and with almost any nonzero activation functions can universally approximate any continuous function on any compact input sets [9,10]. Based on this concept, the extreme learning machine (ELM) algorithm was proposed for batch learning [9,11], which has attracted tremendous attention from various fields for recent years [12–17]. ELM was also extended to semi-supervised/unsupervised tasks [18] and online sequential learning applications (OS-ELM)[19]. Most of these algorithms are based on the principle of least square error minimization, so the performances of these algorithms are easily affected by outliers. In other words, these algorithms are not robust. Inspired by different mechanisms, two robust algorithms were proposed, namely least trimmed squares (LTS) [20–23] and rank-based Wilcoxon neural networks (WNN) [24–26]. LTS and WNN have good generalization capability in outliers existing tasks, but some vital parameters, like learning rate, have to be decided by try and error. In this paper, a new learning machine based on Wilcoxon norm is proposed, then the generalization capability and training speed of both robust and nonrobust algorithms will be compared.

This paper is organized as follows. Section 2 reviews the Wilcoxon neural network proposed by Hsieh [20] and discusses some related problems. Section 3 illustrates the basic background of ELM and discusses the proposed WRELM in detail. The experimental results are conducted in Section 4. Finally, some conclusions are included in Section 5.

* Corresponding author.
E-mail addresses: xiaoliang.xie@ia.ac.cn (X.-L. Xie), guibin.bian@ia.ac.cn (G.-B. Bian), hou@compsys.ia.ac.cn (Z.-G. Hou), zhenqiu.feng@ia.ac.cn (Z.-Q. Feng), jianlong.hao@ia.ac.cn (J.-L. Hao).

http://dx.doi.org/10.1016/j.neucom.2015.12.113
0925-2312/

2. Wilcoxon SLFN

2.1. Wilcoxon norm

The Wilcoxon norm of a vector will be used as the objective function for Wilcoxon learning machines. In order to define the Wilcoxon norm of a vector, a score function is introduced. The score function is a nondecreasing function $\phi : [0,1] \to \mathbb{R}^1$ which satisfies $\int_0^1 \phi(u)\,du = 0$ and $\int_0^1 \phi^2(u)\,du = 1$.

The score $a_\phi(\cdot)$ associated with the score function ϕ is defined by

$$a_\phi(i) = \phi\left(\frac{i}{N+1}\right), \quad i = 1, 2, \ldots, N \tag{1}$$

where N is a fixed positive integer. Hence $a_\phi(1) \le a_\phi(2) \le \ldots \le a_\phi(N)$. It can be shown that the following function is a pseudonorm (seminorm) on $\mathbb{R}^N$:

$$\|e\|_W = \sum_{i=1}^{N} a(R(e_i))e_i = \sum_{i=1}^{N} a(i)e_{(i)} \tag{2}$$

where $e = [e_1, \ldots, e_N]^T \in \mathbb{R}^N$, $R(e_i)$ denotes the rank of e_i among $e_1, \ldots, e_N$, $e_{(1)} \le \ldots \le e_{(N)}$ are the ordered values of $e_1, \ldots, e_N$, $a(i) = \phi[i/(N+1)]$, and $\phi(u) = \sqrt{12}(u - 0.5)$. We call $|e|_W$ defined in Eq. (2) the Wilcoxon norm of the vector e.

It is easy to show that the proposed Wilcoxon norm above satisfies the following properties for a pseudonorm:

(a) $\|e\|_W \ge 0$ for all $e \in \mathbb{R}^N$, if and only if $e_1 = \cdots = e_N$, $|e|_W = 0$.
(b) $\|\alpha e\|_W = |\alpha|\,\|e\|_W$ for all $\alpha \in \mathbb{R}^1$ and $e \in \mathbb{R}^N$.
(c) $|e_1 + e_2|_W \le |e_1|_W + |e_2|_W$ for all $e_1, e_2 \in \mathbb{R}^N$.

2.2. Wilcoxon neural network

In this part, just the core concept of WNN will be illustrated, more details on WNN can refer to [24]. Consider the single-hidden layer Wilcoxon neural network with $n+1$ nodes in its input layer, m nodes in its hidden layer, and p nodes in its output layer.

Let the input vector be $x = [x_1, x_2, \ldots, x_n, 1]^T \in \mathbb{R}^{n+1}$, and let v_{ij} denote the connection from the ith input node to the jth hidden node. The input u_j and output r_j of the jth hidden node are respectively given by

$$u_j = \sum_{i=1}^{n+1} v_{ji}x_i, \quad r_j = f(u_j), \quad \text{for } j = 1, 2, \ldots, m \tag{3}$$

where f is the activation function of hidden nodes.

Let w_{kj} denote the connection weight from the output of the jth hidden node to the kth output node. Then, the output of kth output node t_k and final output y_k are respectively given by

$$t_k = \sum_{j=1}^{m} w_{kj}r_j, \quad y_k = t_k + b_k, \quad \text{for } k = 1, 2, \ldots, p \tag{4}$$

where b_k is the bias of the kth output node.

Assume that the training data set is $\{(\boldsymbol{x}_i, d_i)\}_1^N$ with $\boldsymbol{x}_i \in \mathbb{R}^{n+1}$ and $\boldsymbol{d}_i \in \mathbb{R}^p$, where N is the number of training data, $\boldsymbol{x}_i = [x_{1i}, \ldots, x_{ni}, 1]^T$ is the ith input vector, and $\boldsymbol{d}_i$ is the desired output for the input $\boldsymbol{x}_i$. In the WNN, the approach is to choose network weights ($\boldsymbol{v}$ and $\boldsymbol{w}$) that minimize the Wilcoxon norm of the total residuals of training data

$$D(\boldsymbol{v}, \boldsymbol{w}) = \sum_{k=1}^{p}\sum_{i=1}^{N} a(R(e_{i,k}))e_{i,k} = \sum_{k=1}^{p}\sum_{i=1}^{N} a(i)e_{(i),k} \tag{5}$$

where $e_{i,k} = d_{i,k} - t_{i,k}$, $R(e_{i,k})$ denotes the rank of the residual $e_{i,k}$ among $e_{1,k}, \ldots, e_{N,k}$ and $e_{(1),k} \le \ldots \le e_{(N),k}$ are the ordered values of $e_{1,k}, \ldots, e_{N,k}$.

The neural network used above is the same as the one used in the traditional artificial neural network, except the bias terms at the output node. The main reason is that the Wilcoxon norm is a pseudonorm rather than the usual norm. $\|e\|_W = 0$ implies that $e_1 = \cdots = e_N$, not implies that $e_1 = \cdots = e_N = 0$. Therefore, without the bias terms, the resulting predictive function with small Wilcoxon norm of total residuals may deviate from the desired function by constant offsets. The bias term b_k is estimated by the median of the residuals at the kth output node, i.e., $b_k = \text{med}_{1 \le i \le N}\{d_{ki} - t_{ki}\}$.

The proposed gradient descent based algorithm in [24] can train WNN effectively, however, there is one practical issue involved in real application. The speed of convergence depends highly on the magnitude of the learning rate parameter which is highly task dependant. To guarantee the network convergence, and avoid oscillations during training, the learning rate parameter must be set to a relatively small value, which clearly affects the speed of the algorithm [1]. In this paper, we use an algorithm in linear regression to train WNN, and it will be discussed in the following section.

3. Wilcoxon-norm-based robust extreme learning machine

In this section, a brief description of the ELM algorithm developed by Huang et al. in [9] is given first. Then the WRELM algorithm is introduced.

3.1. ELM algorithm

In supervised batch learning applications, learning algorithms use a finite number of input-output samples for learning networks' parameters. For N arbitrary distinct samples $(x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}^p$, standard SLFNs with m hidden neurons and activation function (or radial basis function) $g(x)$ are modeled as

$$\sum_{j=1}^{m} \boldsymbol{w}_j G(\boldsymbol{a}_j, b_j, \boldsymbol{x}_i) = \boldsymbol{y}_i, \quad \text{for } i = 1, \ldots, N \tag{6}$$

where $\boldsymbol{a}_j$ and b_j are the learning parameters of hidden neurons and $\boldsymbol{w}_j$ is the weight connecting the jth hidden node to output neurons. For additive hidden neuron with the activation function $g(x)$ (e.g., sigmoid or threshold), $G(\boldsymbol{a}_j, b_j, \boldsymbol{x})$ is given in [19]

$$G(\boldsymbol{a}_j, b_j, \boldsymbol{x}) = g(\boldsymbol{a}_j \cdot \boldsymbol{x} + b_j), \quad b_j \in \mathbb{R}. \tag{7}$$

For RBF hidden neuron with Gaussian activation function $g(x)$, $G(a_i, b_i, x)$ is given by $G(\boldsymbol{a}_j, b_j, \boldsymbol{x}) = g\left(\frac{\|\boldsymbol{x} - \boldsymbol{a}_j\|}{2b_j^2}\right)$, $b_j \in \mathbb{R}$.

Eq. (6) can be written compactly as

$$H \cdot W = Y \tag{8}$$

where

$$H = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_m, b_m, \boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_m, b_m, \boldsymbol{x}_N) \end{bmatrix}, W = \begin{bmatrix} \boldsymbol{w}_1^T \\ \vdots \\ \boldsymbol{w}_m^T \end{bmatrix}_{m \times p}$$

and $Y = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix}_{N \times p}$.

H is called the hidden layer output matrix of the network [9]. The ith column of H is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

By minimizing the objective function $\|H \cdot W - Y\|_2^2$, the estimation of output weights of hidden layer can be calculated by

$$W = \arg\min_{\mathbf{w}_k} \|H \cdot W - Y\|_2^2 = H^+ Y \tag{9}$$