Neurocomputing 166 (2015) 164–171

Contents lists available at ScienceDirect

Neurocomputing

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

Extreme learning machine with parallel layer perceptrons

L.D. Tavares [a,*], R.R. Saldanha [a], D.A.G. Vieira [b]

[a] Graduate Program in Electrical Engineering – Federal University of Minas Gerais – Av. Antônio Carlos 6627, 31270-901 Belo Horizonte, MG, Brazil
[b] ENACOM–Handcrafted Technologies – Rua Professor José Vieira de Mendonça, 770, Belo Horizonte Technology Park (BH-TEC), Belo Horizonte, MG, Brazil

ARTICLE INFO

Article history:
Received 13 December 2014
Received in revised form
26 February 2015
Accepted 2 April 2015
Communicated by G.-B. Huang
Available online 20 April 2015

Keywords:
Parallel layer perceptrons
Extreme learning machine
Structural risk minimization
Least square estimate

ABSTRACT

This paper proposes using the Parallel Layer Perceptron (PLP) network, instead of the Single Layer Feedforward neural network (SLFN) in the Extreme Learning Machine (ELM) framework. Differently from the SLFNs which consider cascade layers, the PLP is designed to accomplish also parallel layers, being the SLFN its particular case. This paper explores a particular PLP configuration which considers a nonlinear layer in parallel with a linear layer. For n inputs and m nonlinear neurons, it provides $(n+1)m$ linear parameters, while the SLFN would have only m linear parameters (one for each hidden neuron). Since the ELM is based on adjusting only the linear parameters using the least squares estimate (LSE), the PLP network provides more freedom for the proper adjustment. Results from 12 regression and 6 classification problems are presented considering the training and test errors, the linear vector norm and the system condition number. They point out that the PLP-ELM framework is more efficient than the SLFN-ELM approach.

1. Introduction

Neural networks are mathematical models inspired by the brain functioning which can be applied to a wide range of complex problems, such as nonlinear function approximation, classification, pattern recognition and time-series forecasting. It is well known that a Single-Layer Feedforward Neural (SLFN) network is able to approximate any continuous nonlinear function. This property was verified in 1984 by Cybenco [1] and Funahashi [2], and, more recently, Huang and Babri [3] demonstrated its learning upper bounds. However, the SFLN training time was its major bottleneck since most algorithms were based on gradient descent methods (or similar), such as the backpropagation [4].

In order to tackle this bottleneck, the Extreme Learning Machine (ELM) approach is gaining researchers attention since the pioneer Huang et al.' work [5–7]. The ELM is an SLFN training method that simplifies the training processes. The SLFN-ELM differs from the traditional neural networks training methods by analytically determining the linear parameters by means of linear least squares estimate (LSE) solutions.

In this sense, the SFLN-ELM may be interpreted as a linear system problem in the form $\mathbf{H}\beta = Y_d$, where $\mathbf{H}$ is its hidden layer, Y_d is the desired output vector and β are the linear parameters [6]. Classical SFLN-ELM approaches do not adjust the nonlinear hidden layer parameters.

There are several advantages in the SLFN-ELM approach, which can be highlighted: it considers the minimum norm of the output layer weights, the training is extremely fast (when compared to the other conventional gradient descent learning algorithms), it requires fewer parameter settings (the SLFN-ELM only adjust the output weights), and it presents a good generalization performance [7].

Several studies were, and are still performed, about SLFN-ELMs. Output weights regularization (relative to the error or other criteria) and multi-objective approaches were studied in [8,9]. The determination of the hidden layer neurons number (which can also be considered a multi-objective problem) was examined in many studies, as in [10–16], however, it remains an open problem. Although the SLFN-ELM initializes the hidden layer with random weights, some studies indicate other initialization methods, such as [17,18]. SLFN-ELM uses LSE solutions to determine the output layer weights, however, some evolutionary strategies have been successfully applied, such as [19–23]. Kernel-based architectures (similar to Support Vector Machines – SVM) were also experimented in the papers [24–28] and fuzzy-like systems in [29]. Although originally ELM is a single layer network, the use of multiple layers was discussed in [30]. Some others SLFN-ELM theoretical issues, for instance, the feasibility and generalization performance, are discussed in [31,32].

Many applications have employed SLFN-ELM, for instance: fast object recognition and image classification [33–36]; credit risk evaluation [37], health [38], big data classification [39], use of priori knowledge [40], recommendation systems [41], and many others. Theoretical and practical trends related to ELMs can be found in [42].

* Corresponding author.
E-mail addresses: tavares@dcc.ufmg.br (L.D. Tavares), rodney@cpdee.ufmg.br (R.R. Saldanha).

http://dx.doi.org/10.1016/j.neucom.2015.04.018
0925-2312/

As discussed in [43], the neural network training process is, naturally, a multi-objective problem where it wants to balance the empirical risk and complexity. Thus, the SLFN-ELM training method performs the empirical risk minimization, by solving the $\|\mathbf{H}\beta - Y_d\|_2$ problem, and minimizing the output weights norm $\|\beta\|_2$. Therefore, it maps the minimal empirical risk extreme point such that the output weights norm is minimized. As presented in [44] the weights norm is more important for generalization than the network size, reducing the structural risk [45].

It is possible to observe that the norm of $\|\beta\|_2$ is strongly influenced by the **H** matrix composition and its properties. An ill-conditioned **H** matrix will certainly lead to worse results of $\|\beta\|_2$ and the $\mathbf{H}\beta = Y_d$ system will become ill-formed, as a whole. Reflecting on these characteristics, some questions arise (i) Is it possible to modify the hidden layer structure in order to benefit $\|\beta\|_2$? (ii) Is it possible to keep, in this modified structure, a well-conditioned hidden layer space?

In 2003, Caminhas et al. proposed a machine learning where the model output is computed in parallel layers, called Parallel Layer Perceptron (PLP) [46]. The PLP is based on parallel characteristics of Adaptive-Network-based Fuzzy Inference System (ANFIS) [47], and had shown excellent results in terms of generalization ability and training speed, even using traditional learning methods based on gradients or hybrid algorithms. Another important feature presented by the PLP model is that it builds a well-conditioned hidden layer and keeps under control the norm of the linear weight parameters naturally [43,48]. The PLP has been used in many problems, such as cirurgical hand–eyes coordination [49], ground penetrating radar inverse problems [50–53], noise filtering [54], among others. Differently from the SLFN, which has only one linear parameter for each hidden neuron, the PLP can have as much as desired by the user. This paper explores a simple linear parallel layer which has $(n+1)m$ free linear parameters per neuron, where n is the number of inputs and m is the number of neurons per parallel layer.

This work aims at using the PLP architecture in conjunction with the ELM training method, forming the Parallel Layer Perceptron-Extreme Learning Machine (PLP-ELM). It is expected to build a parallel learning machine which has the good characteristics of both models: good generalization, high speed, and low model complexity.

The rest of the paper is organized as follows: Section 2 details the proposed PLP-ELM mathematical model and Section 3 describes its training method. The performance evaluation and comparisons with ELM and other models are presented in Section 4. The PLP-ELM has presented better results in most of the 18 tested data sets, considering the training and test errors, as well as the linear vector norm and nonlinear matrix condition number. Finally, the Section 5 presents the conclusions and some future works.

2. Parallel Layer Perceptron Extreme Learning Machine mathematical model

The Parallel Later Perceptron output, Y_t, considering the input $\mathbf{x}_t$ and m hidden perceptron per layer is computed as [46,48]

$$Y_t = \beta\left(\sum_{j=1}^{m}\left[\gamma(a_{jt})\phi(b_{jt})\right]\right) \tag{1}$$

where a_{jt} and b_{jt} are

$$a_{jt} = \sum_{i=1}^{n+1} p_{ji}x_{it} \tag{2}$$

$$b_{jt} = \sum_{i=1}^{n+1} v_{ji}x_{it} \tag{3}$$

and $\beta(\cdot)$, $\gamma(\cdot)$ and $\phi(\cdot)$ are activation functions (hyperbolic tangent, Gaussian, linear, etc.), p_{ji} and v_{ji} are components of **P** and **V** weights matrices, x_{it} is the ith input for tth sample, x_{0t} is the perceptron bias and Y_t is the tth position of Y output vector. The Y_t dimension depends only on the β function.

Fig. 1 shows the PLP-ELM architecture and how the layers are arranged in parallel form.

Just as the traditional SLFNs, all weights may be adjusted during the training phase, however, some distinctions between SLFN and PLP must be highlighted. Firstly, in SLFN approach the input–output mapping is made using a function of functions and in the PLP approach this is done by applying the product of functions (although the PLP topology also allows function of functions arrangement). Moreover, the PLP topology simplifies the implementation in parallel machines or clusters.

As a particular case of Eq. (1), assume that the activation functions $\beta(\cdot)$ and $\gamma(\cdot)$ are linear. In this case, the network output Y_t is computed as

$$Y_t = \sum_{j=1}^{m}\left[a_{jt}\phi(b_{jt})\right] \tag{4}$$

Replacing Eqs. (2) and (3) in (4)

$$Y_t = \sum_{j=1}^{m}\left[\left(\sum_{i=1}^{n+1} p_{ji}x_{it}\right)\phi\left(\sum_{i=1}^{n+1} v_{ji}x_{it}\right)\right] \tag{5}$$

The particular case described in Eq. (4) has some desirable characteristics. The error surface in relation to p_{ji}, which is the linear parameter, is a quadratic structure, hence a least squares estimate solution can be easily used.

The number of linear parameters in the PLP configuration is nm higher than in the case of cascade single layer perceptrons. This feature gives more flexibility to the least-squares, as will be demonstrated in the experiments.

3. Proposed PLP-ELM training method

The PLP-ELM training method is based on the adjustment of only the linear terms of Eq. (5), using the least squares estimate (LSE) solution. This is possible once the output Y_t is a linear function of parameters p_{jk}. The nonlinear parameters are initialized randomly and there is no need to adjust it during the training process.

Consider $\ell_k = p_{ji}$, where $k = n(j-1)+i$, where $j = 1,\ldots,m$ and $i = 0,\ldots,n+1$ thus ℓ is the transformation of the matrix **P** into a column vector, as

$$\ell = [p_{11}\ldots p_{1(n+1)}p_{21}\ldots p_{2(n+1)}p_{h1}\ldots p_{h(n+1)}]' \tag{6}$$

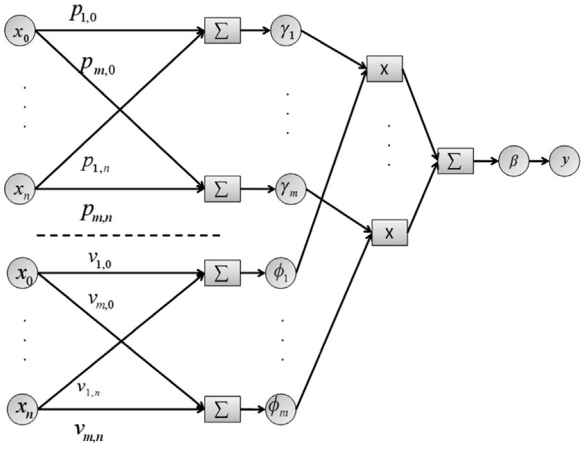

Fig. 1. Parallel Layer Perceptron Extreme Learning Machine architecture.