Applied Soft Computing xxx (2018) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

A non-iterative method for pruning hidden neurons in neural networks with random weights

Pablo A. Henríquez [a], Gonzalo A. Ruz [a,b,*]

[a] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Av. Diagonal Las Torres 2640, Peñalolén, Santiago, Chile
[b] Center of Applied Ecology and Sustainability (CAPES), Santiago, Chile

ARTICLE INFO

Article history:
Received 29 September 2016
Received in revised form 5 March 2018
Accepted 10 March 2018
Available online xxx

Keywords:
Non-iterative learning
Neural networks
Random weights
Garson's algorithm
Pruning
Regression
Classification

ABSTRACT

Neural networks with random weights have the advantage of fast computational time in both training and testing. However, one of the main challenges of single layer feedforward neural networks is the selection of the optimal number of neurons in the hidden layer, since few/many neurons lead to problems of underfitting/overfitting. Adapting Garson's algorithm, this paper introduces a new efficient and fast non-iterative algorithm for the selection of neurons in the hidden layer for randomization based neural networks. The proposed approach is divided into three steps: (1) train the network with *h* hidden neurons, (2) apply Garson's algorithm to the matrix of the hidden layer, and (3) perform pruning reducing hidden layer neurons based on the harmonic mean. Our experiments in regression and classification problems confirmed that the combination of the pruning technique with these types of neural networks improved their predictive performance in terms of mean square error and accuracy. Additionally, we tested our proposed pruning method with neural networks trained under sequential learning algorithms, where Random Vector Functional Link obtained, in general, the best predictive performance compared to online sequential versions of extreme learning machines and single hidden layer neural network with random weights.

1. Introduction

Single layer feedforward neural networks (SLFNs) with random weights, as random basis function approximators, have received considerable attention in classification and regression problems because of their universal approximation capability [1–3]. Special features of these learner models come from weights specification, that is, the input weights and biases are randomly assigned and the output weights can be analytically evaluated by a Moore–Penrose generalized inverse of the hidden output matrix.

Neural networks with random weights have many advantages in comparison to other iterative methods of learning, such as back-propagation (BP) [4] or support vector machine (SVM) [5]. The first advantage is the computational time required to train the model, being much lower in comparison to iterative methods. A second advantage is that the user only needs to specify the number of hidden neurons. Nevertheless, this parameter in many cases has a significant impact on the performance of the model, therefore a method to assist in the selection of the architecture of SLFNs is desirable. Typically, the number of neurons in the hidden layer is pre-determined by a trial-and-error approach [6]. It is known that neural networks with a small hidden layer will not be capable of generalizing the training data (underfitting); on the other hand, if the network has a large number of hidden neurons, this can give place to overfitting. Therefore, the number of hidden neurons plays a crucial role in an appropriate training of SLFNs. To address this issue, research work has focused primarily on determining the network's architecture within the learning algorithm. This approach consists in the pruning of unnecessary neurons during the training process [7,8].

In the last decades, several methods to determine the contribution of each input variable (attribute) have been proposed, such as the PaD method [9], Perturb method [10], and the Profile method [11]. In the literature there are many methods for the assessment of the relative importance of the independent variables. Garson developed an algorithm based on the connection weights of a neural network in order to determine the relative importance of each input variable [12]. This same idea was modified and applied by Goh [13].

* Corresponding author at: Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Av. Diagonal Las Torres 2640, Peñalolén, Santiago, Chile.
E-mail addresses: pabhenriquez@alumnos.uai.cl, pablohenriquez8386@gmail.com (P.A. Henríquez), gonzalo.ruz@uai.cl (G.A. Ruz).

https://doi.org/10.1016/j.asoc.2018.03.013
1568-4946/

As mentioned, the above methods are concerned with the importance of the input variables (for feature selection). Instead, we consider in this work, the determination of the relative importance of neurons in the hidden layer for pruning purposes. In particular, due to its non-iterative nature, simplicity, and effectiveness, as shown in [14–17], we adapt Garson's algorithm for pruning hidden nodes in neural networks with random weights.

The contributions of this paper are as follows:

- We propose a non-iterative method for pruning hidden neurons for a single hidden layer neural network with random weights (RWSLFN), Random Vector Functional Link Network (RVFL), Extreme Learning Machines (ELM), and randomized neural networks in general.
- The proposed method is based on Garson's algorithm, which is commonly used for feature selection (input layer) in the context of neural networks, but not for pruning hidden nodes.
- We evaluate and compare the performance and training time of the proposed method for regression and classification problems, considering iterative and non-iterative methods in neural networks with random weights.

The paper is organized as follows. Section 2 presents non-iterative learning algorithms for neural networks with random weights. The proposed method called Garson-pruned (GP) is introduced in Section 3. Section 4 describes our experimental setup and procedures. In Section 5, we compare the performance using different thresholds. The results in regression and classification problems using the proposed method and comparisons with other algorithms is presented in Section 6. Section 7 describes the proposed method applied to sequential algorithms. And finally, in Section 8, we draw the main conclusions from our work.

2. Non-iterative learning algorithms for neural networks with random weights

In the literature, there are many kinds of randomization of the inner weights for feedforward neural networks. A schematic diagram of an SLFN with random weights is shown in Fig. 1. In 1992, for instance, Schmidt et al. [18] proposed a single hidden layer neural network with random weights (RWSLFN) assignments between the input and hidden layers and least-square estimation on the output weights as the training method. In 1994, Pao et al. [19] proposed a functional link neural network named Random Vector Functional Link Network (RVFL). In 1996, Chen et al. [20] found the maximum number of hidden nodes to be $N-r-1$ for RVFL network with a constant bias to learn a mapping within a given precision based on an n-dimensional N-pattern dataset, where r is the rank of the dataset. Furthermore, a robust weighted least-square method is proposed in order to eliminate outliers. In 1998, Broomhead et al. [21] studied the implicit assumptions made when employing a feedforward layered network model to analyze complex data. The use of radial basis function networks (RBF) is considered by Lowe [22] and Park et al. [23]. In particular, this latter work offers a proof that RBF networks with one hidden layer are capable of universal approximation. Hornik [24] investigated the approximation ability of standard multilayer feedforward networks with as few as one hidden layer. As shown by Hornik, when the activation function is bounded and nonconstant, the standard multilayer feedforward networks are universal approximators. Recently, Zhang and Suganthan [25] conducted a survey on randomized algorithms for training neural networks. Among the different machine learning techniques, a modified version of RWSLFN called extreme learning machine (ELM), proposed by Huang et al. [26] in 2004, has gained attention in the past few years [27].

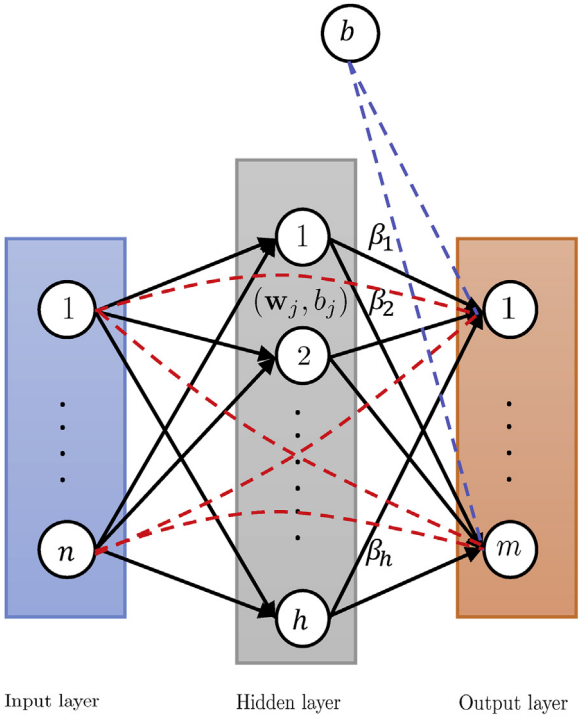

Fig. 1. Schematic diagram of an SLFN with random weights. Dashed red arrows represent direct connections between input and output neurons. Dashed blue arrows show the bias in the output node. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

The main procedure in non-iterative learning algorithms consists in randomly fixing parameters in the model's configurations, rather than tuning them via a time-consuming iterative process (such as BP [4] or SVM [5]).

Consider a training set containing N arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{o}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ and $\mathbf{o}_i = [o_{i1}, o_{i2}, \ldots, o_{im}]^T \in R^m$, with h hidden layer nodes and the activation function $g(x)$. In general, a non-iterative algorithm can be defined as

$$\hat{\mathbf{o}}_i = \sum_{j=1}^{h} \boldsymbol{\beta}_j g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) + \mathbf{b} + \sum_{j=h+1}^{h+n} \boldsymbol{\beta}_j \cdot \mathbf{x}_i,\ i = 1, \ldots, N, \tag{1}$$

where $\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T$ is the weight vector connecting the jth hidden node with the input nodes. $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jm}]^T$ is the weight vector connecting the jth hidden node with the output nodes; b_j is the threshold of the jth hidden node; h represents the number of hidden neurons; $\mathbf{b}$ is a bias vector of length m in the output neurons.

Eq. (1) can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \hat{\mathbf{O}}, \tag{2}$$

where $\mathbf{H}$ is the hidden layer output matrix of the neural network, computed as follows:

$$\mathbf{H}(\mathbf{w}_j, b_j, \mathbf{x}_i) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_h \cdot \mathbf{x}_1 + b_h) & b & \mathbf{x}_1^T \\ \vdots & \vdots & \vdots & \vdots & \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_h \cdot \mathbf{x}_N + b_h) & b & \mathbf{x}_N^T \end{bmatrix}_{N \times (h+1+n)}, \tag{3}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & \ddots & \vdots \\ \beta_{(h+1+n)1} & \cdots & \beta_{(h+1+n)m} \end{bmatrix}_{(h+1+n) \times m}, \tag{4}$$

Please cite this article in press as: P.A. Henríquez, G.A. Ruz, A non-iterative method for pruning hidden neurons in neural networks with random weights, Appl. Soft Comput. J. (2018), https://doi.org/10.1016/j.asoc.2018.03.013