Neurocomputing 167 (2015) 158–164

Contents lists available at ScienceDirect

Neurocomputing

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

Brief Papers

Extreme learning machine based supervised subspace learning

Alexandros Iosifidis

Department of Signal Processing, Tampere University of Technology, Finland

ARTICLE INFO

Article history:
Received 5 November 2014
Received in revised form
23 February 2015
Accepted 29 April 2015
Communicated by Ning Wang
Available online 22 May 2015

Keywords:
Extreme Learning Machine
Supervised subspace learning
Network targets calculation

ABSTRACT

This paper proposes a novel method for supervised subspace learning based on Single-hidden Layer Feedforward Neural networks. The proposed method calculates appropriate network target vectors by formulating a Bayesian model exploiting both the labeling information available for the training data and geometric properties of the training data, when represented in the feature space determined by the network's hidden layer outputs. After the calculation of the network target vectors, Extreme Learning Machine-based neural network training is applied and classification is performed using a Nearest Neighbor classifier. Experimental results on publicly available data sets show that the proposed approach consistently outperforms the standard ELM approach, as well as other standard methods.

1. Introduction

Extreme Learning Machine (ELM) is an algorithm that has been successfully applied for Single-hidden Layer Feedforward Neural (SLFN) networks training, leading to fast network training with low human supervision [1]. The main idea in ELM-based approaches is that the network's hidden layer weights and bias values need not to be learned and can be determined by random assignment. By adopting either the squared loss [2] or the hinge loss [3] of the prediction error, the network output weights are, subsequently, analytically calculated. This approach is not in line with conventional SLFN network training approaches, like the Back-propagation [4] and the Levenberg–Marquardt [5] algorithms, where both the network's hidden and output parameters are adjusted by following an optimization process, e.g., by applying gradient descend-based optimization. However, despite the random hidden layer parameters determination, it has been proven that SLFN networks trained by using the ELM algorithm have the properties of global approximators [6,7]. ELMs not only tend to reach the smallest training error, but also the smallest output weight norm. For feedforward networks reaching a small training error, smaller output weight norm results in better generalization performance [8]. Due to its effectiveness and its fast learning process, the ELM network has been adopted in many classification problems and many ELM variants have been proposed in the last few years, extending the ELM network properties along different directions [9–13].

While ELM-based techniques have proven their efficiency and effectiveness in supervised [9,10,14] and semi-supervised [12,15,16] classification, their application in subspace learning has not been well investigated. An attempt to exploit the ELM approach in unsupervised subspace learning has recently been proposed [16], where it has been shown that ELM-based unsupervised subspace learning is able to achieve excellent performance, and even outperform several state-of-the-art unsupervised subspace learning methods. The application of the ELM approach in supervised subspace learning has not been investigated yet. This is probably due to the fact that previous works have focused their attention either on designing good optimization problems for the calculation of the network output weights, e.g. [2,3,14], or to exploit Linear Algebra theory in order to determine a good set of input parameters, e.g. [17–19]. However, for the design of a supervised subspace learning technique based on regression models, one should also focus on the determination of appropriate target vectors. This paper focuses on the determination of such network target vectors.

In more detail, in this paper, we propose an ELM-based supervised subspace learning method that follows a Bayesian approach. We focus our attention on designing appropriate network target vectors that are able to capture both the class information that is available for the training data and training data geometric properties when represented in the feature space determined by the network's hidden layer outputs, noted as ELM space hereafter. This is achieved by formulating a Bayesian model that takes into account the ELM assumptions, which are exploited in order to determine a convenient class representation in the ELM space. Subsequently, the network can be trained by using any of the methods proposed for supervised ELM network training [1,2,20,19]. After the determination of the non-linear data mapping function, classification in the feature space of reduced dimensionality is performed by using the Nearest Neighbor classifier. We evaluate the proposed approach in standard and real classification problems, where we compare its performance with that of the standard ELM approach, as well as with Kernel Spectral Regression [21] and Support Vector Machine [22] based classification.

E-mail address: alexandros.iosifidis@tut.fi

http://dx.doi.org/10.1016/j.neucom.2015.04.083
0925-2312/

The rest of the paper is organized as follows. We provide an overview of ELM-based SLFN network training in Section 2. The proposed approach for ELM network target vectors calculation is described in Section 3. Experimental results evaluating the performance of the proposed approach in standard and real classification problems are provided in Section 4. Finally, conclusions are drawn in Section 5.

2. Previous work

In this section, we describe the ELM, Regularized ELM (RELM) and Kernel ELM (KELM) algorithms proposed in [1,2]. Subsequently, we briefly discuss extensions of the ELM approach that have been recently proposed for supervised and semi-supervised SLFN network training.

Let us denote by $\{\mathbf{x}_i, c_i\}_{i=1}^N$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class labels $c_i \in \{1, ..., C\}$. These vectors are employed in order to train a SLFN network consisting of D input (equal to the dimensionality of $\mathbf{x}_i$), L hidden and C output (equal to the number of classes involved in the classification problem) neurons. The number of hidden layer neurons is a parameter of the algorithm and is usually selected to be much greater than the number of classes, i.e., $L \gg C$ [2,14]. In ELM-based approaches, the network's input weights $\mathbf{W}_{in} \in \mathbb{R}^{D\times L}$ and the hidden layer bias values $\mathbf{b} \in \mathbb{R}^L$ are randomly assigned, while the network's output weights $\mathbf{W}_{out} \in \mathbb{R}^{L\times C}$ are analytically calculated.

Let us denote by $\mathbf{v}_j$, $\mathbf{w}_k$ and w_{kj} the jth column of $\mathbf{W}_{in}$, the kth row of $\mathbf{W}_{out}$ and the jth element of $\mathbf{w}_k$, respectively. Given an activation function $\Phi(\cdot)$ for the network hidden layer neurons and using a linear activation function for the network output layer neurons, the response $\mathbf{o}_i = [o_{i1}, ..., o_{iC}]^T$ of the network corresponding to $\mathbf{x}_i$ is calculated by

$$o_{ik} = \sum_{j=1}^{L} w_{kj}\, \Phi(\mathbf{v}_j, b_j, \mathbf{x}_i), \quad k = 1, \ldots, C. \tag{1}$$

It has been shown [7,23,2] that, almost any nonlinear piecewise continuous activation function $\Phi(\cdot)$ can be used for the calculation of the network hidden layer outputs, like the sigmoid, sine, Gaussian, hard-limiting and Radial Basis Function (RBF), Fourier series, etc. By storing the network hidden layer outputs $\boldsymbol{\phi}_i \in \mathbb{R}^L$ corresponding to all the training vectors $\mathbf{x}_i, i = 1, ..., N$ in a matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_N]$, or

$$\boldsymbol{\Phi} = \begin{bmatrix} \Phi(\mathbf{v}_1, b_1, \mathbf{x}_1) & \cdots & \Phi(\mathbf{v}_1, b_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ \Phi(\mathbf{v}_L, b_L, \mathbf{x}_1) & \cdots & \Phi(\mathbf{v}_L, b_L, \mathbf{x}_N) \end{bmatrix}, \tag{2}$$

Eq. (1) can be expressed in a matrix form as

$$\mathbf{O} = \mathbf{W}_{out}^T \boldsymbol{\Phi}, \tag{3}$$

where $\mathbf{O} \in \mathbb{R}^{C\times N}$ is a matrix containing the network responses for all training data $\mathbf{x}_i$.

Standard ELM algorithm [1] assumes zero training error. That is, it is assumed that $\mathbf{o}_i = \mathbf{t}_i, i = 1, ..., N$, or by using a matrix notation $\mathbf{O} = \mathbf{T}$, where $\mathbf{T} = [\mathbf{t}_1, ..., \mathbf{t}_N]$ is a matrix containing the network target vectors. By using (3), the network output weights $\mathbf{W}_{out}$ are analytically calculated by

$$\mathbf{W}_{out} = \boldsymbol{\Phi}^{\dagger}\, \mathbf{T}^T, \tag{4}$$

where $\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1}\boldsymbol{\Phi}$ is the generalized pseudo-inverse of $\boldsymbol{\Phi}^T$. After the calculation of the network output weights $\mathbf{W}_{out}$, the response of the corresponding to a vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$\mathbf{o}_t = \mathbf{W}_{out}^T \boldsymbol{\phi}_t, \tag{5}$$

where $\boldsymbol{\phi}_t = [\Phi(\mathbf{v}_1, b_1, \mathbf{x}_t), ..., \Phi(\mathbf{v}_L, b_L, \mathbf{x}_t)]^T$ is the network hidden layer output for $\mathbf{x}_t$.

The calculation of the network output weights $\mathbf{W}_{out}$ through (4) is sometimes inaccurate, since the matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is singular in the case where $L > N$. A regularized version of the ELM algorithm that allows small training errors and tries to minimize the norm of the network output weights $\mathbf{W}_{out}$ has been proposed in [2], where the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize}: \quad \mathcal{J} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{6}$$

$$\textbf{Subject to}: \quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad : i = 1, \ldots, N, \tag{7}$$

where $\boldsymbol{\xi}_i \in \mathbb{R}^C$ is the error vector corresponding to $\mathbf{x}_i$ and $c > 0$ is a parameter denoting the importance of the training error in the optimization problem. Based on the Karush–Kuhn–Tucker (KKT) theorem [24], the network output weights $\mathbf{W}_{out}$ can be determined by solving the dual optimization problem

$$\tilde{\mathcal{J}} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 - \sum_{i=1}^{N}\mathbf{a}_i\left(\mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i + \boldsymbol{\xi}_i\right), \tag{8}$$

which is equivalent to (6). By calculating the derivatives of $\tilde{\mathcal{J}}$ with respect to $\mathbf{W}_{out}$, $\boldsymbol{\xi}_i$ and $\mathbf{a}_i$ and setting them equal to zero, the network output weights $\mathbf{W}_{out}$ are obtained by

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \frac{1}{c}\mathbf{I}\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T, \tag{9}$$

or

$$\mathbf{W}_{out} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{T}^T = \boldsymbol{\Phi}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{T}^T, \tag{10}$$

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *ELM kernel matrix*, having elements equal to $[\mathbf{K}]_{i,j} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_j$ [25]. In [2] it has been also shown that, by exploiting the kernel trick [26–28], $\mathbf{K}$ can be any positive semidefinite kernel matrix defined over the input data $\mathbf{x}_i$. By using (10), the response of the network for a vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$\begin{aligned}\mathbf{o}_t &= \mathbf{W}_{out}^T\boldsymbol{\phi}_t = \mathbf{T}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{\phi}_t \\ &= \mathbf{T}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{k}_t,\end{aligned} \tag{11}$$

where $\mathbf{k}_t \in \mathbb{R}^N$ is a vector having elements equal to $\mathbf{k}_{t,i} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_t$.

Several extensions of the ELM algorithm have been proposed in the literature for supervised and semi-supervised classification [20,3,11,29,12,19]. All approaches try either to design good optimization problems for $\mathbf{W}_{out}$ calculation, or to exploit Linear Algebra theory in order to determine a good set of input parameters $\{\mathbf{W}_{in}, \mathbf{b}\}$. In the case of supervised classification, the elements of the network target vectors $\mathbf{t}_i = [t_{i1}, \ldots, t_{iC}]^T$, each corresponding to a training vector $\mathbf{x}_i$, are set to $t_{ik} = 1$ for vectors belonging to class k, i.e., when $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. In the case of semi-supervised classification, the target vectors of the labeled training vectors are set as above, while the target vectors of the unlabeled training vectors are set equal to zero. That is, for the determination of the network target vectors, only the labeling information of the training data is exploited. In the following, we propose a method for the determination of the network target vectors that takes into account both the training data labels and training data geometric properties when represented in the ELM space.