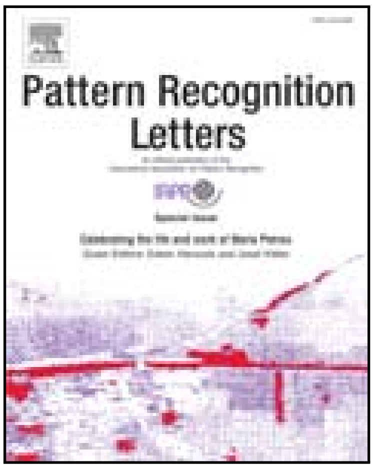

Sparse Extreme Learning Machine classifier exploiting Intrinsic Graphs

Alexandros Iosifidis, Anastasios Tefas, Ioannis Pitas

PII: S0167-8655(15)00245-7
DOI: 10.1016/j.patrec.2015.07.036
Reference: PATREC 6305

To appear in: *Pattern Recognition Letters*

Received date: 6 March 2015
Accepted date: 24 July 2015

Highlights

- We propose an optimization scheme for Sparse Extreme Learning Machine.
- The proposed method exploits geometric data information of intrinsic graphs.
- The proposed approach enhances classification performance.

Pattern Recognition Letters
journal homepage: www.elsevier.com

Sparse Extreme Learning Machine classifier exploiting Intrinsic Graphs

Alexandros Iosifidis[a,**], Anastasios Tefas[b], Ioannis Pitas[b]

[a]*Department of Signal Processing, Tampere University of Technology, FIN-33720, Tampere, Finland*
[b]*Department of Informatics, Aristotle University of Thessaloniki, 54124, Thessaloniki, Greece*

ARTICLE INFO

Article history:

2000 MSC:
41A05
41A10
65D05
65D17

Keywords:
Sparse Extreme Learning Machine
Intrinsic Graphs

ABSTRACT

This paper presents an analysis of the recently proposed sparse Extreme Learning Machine (S-ELM) classifier and describes an optimization scheme that can be used to calculate the network output weights. This optimization scheme exploits intrinsic graph structures in order to describe geometric data relationships in the so-called ELM space. Kernel formulations of the approach operating in ELM spaces of arbitrary dimensions are also provided. It is shown that the application of the optimization scheme exploiting geometric data relationships in the original ELM space is equivalent to the application of the original S-ELM to a transformed ELM space. The experimental results show that the incorporation of geometric data relationships in S-ELM can lead to enhanced performance.

1. Introduction

In the training of Extreme Learning Machine (ELM)-based single hidden layer feedforward neural (SLFN) networks, the network hidden layer parameters are randomly assigned while the network output parameters are, subsequently, analytically calculated. Similar approaches have been also shown to be efficient in several neural network training methods (Broomhead and Lowe (1988); Schmidt et al. (1992); Pao et al. (1994); Chen (1996); Widrow et al. (2013)), as well as in other learning processes (Rahimi and Recht (2008)). Algorithms following this approach assume that the learning processes used to determine the hidden layer and the output weights need not be connected. In addition, it is assumed that the network hidden layer weights can be randomly assigned, therefore defining a random (nonlinear) mapping of the input space to a new (usually high-dimensional) feature space. The problem to be solved can be transformed to a linear problem in the new feature space by using a large number of (independent) hidden layer weights; thus, linear techniques such as mean square estimation can be used to determine the network's output weights. The fact that the network's hidden and output weights are determined independently has a number of advantages that can be exploited, for example, for facilitating the implementation of parallel/distributed systems. In addition, it has been shown to perform well in many classification problems.

In the original ELM algorithm (Huang et al. (2004)), the trained network tends to reach not only the smallest training error but also the smallest output weight norm. For networks reaching small training errors, smaller output weight norms indicate better generalization performance (Bartlett (1998)). Since its first proposal (Huang et al. (2004)), several optimization schemes have been proposed to calculate the network output parameters, each highlighting different properties of the ELM networks (Li et al. (2005); Liang et al. (2006); Huang et al. (2006); Huang and Chen (2008); Feng et al. (2009); Miche et al. (2010); Wang et al. (2011); Huang et al. (2012); Iosifidis et al. (2013); Bai et al. (2014); Iosifidis et al. (2014)). Although the determination of the hidden layer network outputs is based on randomly assigned input parameters, it has been shown that SLFN networks trained by the ELM algorithm have the properties of global approximators (Huang et al. (2006); Liu et al. (2015a,b)). In addition, it has been shown that ELM networks are able to outperform other sophisticated classification schemes, such as the support Vector Machine classifier (Huang

**Corresponding author: Tel: +358404794688;
e-mail: `alexandros.iosifidis@tut.fi` (Alexandros Iosifidis)

et al. (2012); Bai et al. (2014); Huang (2014)).

Recently, an optimization scheme exploiting the hinge loss of the training error for calculating the network output weights has been proposed (Bai et al. (2014)). It exploits the fact that the network output weights can be expressed as a sparse representation of the training data representations in the feature space determined by the hidden network outputs. Thus, testing in both the original and kernel ELM formulations exploiting the hinge loss of the training error is faster than the calculation of the network output weights exploiting the squared loss of the training error. In order to speed up the training process of the so-called sparse ELM (S-ELM) networks, a sequential minimal optimization (SMO)-based optimization algorithm has also been proposed by Bai et al. (2014). By exploiting such an optimization approach, it has been shown that S-ELM is both effective and efficient. Experimental results show that it is able to outperform ELM formulations by exploiting the squared loss of the training error, while its training and test computational costs are lower than those of ELMs and SVMs (Bai et al. (2014)).

In this paper, we describe an optimization scheme for S-ELM-based SLFN network training, which exploits intrinsic graph structures expressing class geometric relationships of the training data in the feature space determined by the network hidden layer outputs, noted as ELM space hereafter. This optimization scheme is also extended to exploiting intrinsic graph structures that express class geometric relationships of the training data in arbitrary-dimensional ELM spaces used in kernel ELM formulations (Bai et al. (2014); Iosifidis et al. (2015)). Intrinsic graphs have also been exploited in SVM-based classification (Orphanidis and Tefas (2012); Arvanitidis and Tefas (2012)) and ELM networks using the squared loss of the training error (Iosifidis et al. (2013, 2014)). Here, the use of such an approach for S-ELM network training is also shown. It is shown that S-ELM networks trained by applying the adopted optimization scheme achieve better classification performance compared with S-ELM networks trained by applying the original optimization scheme, as described by Bai et al. (2014). In addition, in order to exploit fast optimization algorithms like those proposed by Bai et al. (2014) and Sha et al. (2007), the application of the adopted optimization scheme on the original (kernel) ELM space is shown to be equivalent to the application of the original S-ELM optimization scheme to a transformed (kernel) ELM space.

The rest of the paper is structured as follows: In Section 2, an overview of the S-ELM algorithm is provided. In Section 3, an optimization scheme is described for S-ELM-based network training, which exploits geometric data information described in intrinsic graphs. Experiments comparing the performance of S-ELM with that of our optimization scheme are described in Section 5. Finally, conclusions are drawn in Section 6.

2. Overview of S-ELM networks

Let us denote by $\{\mathbf{x}_i, l_i\}_{i=1,...,N}$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class labels $l_i \in \{1, \dots, C\}$. We would like to employ $\{\mathbf{x}_i, l_i\}_{i=1,...,N}$ in order to train a SLFN network using the S-ELM algorithm (Bai et al. (2014)). Such a network consists 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 L is a parameter of the S-ELM algorithm, and it is usually set to be much greater than the number of classes C, that is, $L \gg C$. The elements of the network target vectors $\mathbf{t}_i = [t_{i1}, ..., t_{iC}]^T$, each corresponding to a training vector $\mathbf{x}_i$, are set to $t_{ik} = 1$ for vectors belonging to class k, that is, when $l_i = k$, and to $t_{ik} = -1$ when $l_i \neq k$. In S-ELMs, the network 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 output weights $\mathbf{W}_{out} \in \mathbb{R}^{L \times C}$ are analytically calculated, as subsequently described.

Given an activation function $\Phi(\cdot)$ for the network hidden layer and using a linear activation function for the network output layer, the response $\mathbf{o}_i = [o_{i1}, \dots, o_{iC}]^T$ of the network corresponding to $\mathbf{x}_i$ is calculated by

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

where $\mathbf{v}_j$ is the j-th column of $\mathbf{W}_{in}$, $\mathbf{w}_k$ is the k-th column of $\mathbf{W}_{out}$, and w_{kj} is the j-th element of $\mathbf{w}_k$. It has been shown that almost any nonlinear piecewise continuous activation functions $\Phi(\cdot)$ can be used to calculate the network hidden layer outputs, for example, the sigmoid, polynomial, Radial Basis Fnction (RBF), RBF-χ^2, and Fourier series (Huang et al. (2006); Huang and Chen (2008); Huang et al. (2012); Iosifidis et al. (2013)). It has also been recently proven that S-ELMs using polynomials, Nadaraya–Watson, and sigmoid functions attain the theoretical generalization bound of feedforward neural networks. For the remaining activation function choices, the Tikhonov regularization can be applied to guarantee the weak regularity of the hidden layer output matrix while not sacrificing the network's generalization capability (Liu et al. (2015a,b)).

By storing the network hidden layer outputs $\boldsymbol{\phi}_i \in \mathbb{R}^L$ corresponding to all the training vectors $\mathbf{x}_i$, $i = 1, \dots, N$ in a matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N]$, the network response for all the training data $\mathbf{O} \in \mathbb{R}^{C \times N}$ can be expressed in a matrix form as follows:

$$\mathbf{O} = \mathbf{W}_{out}^T \boldsymbol{\Phi}. \tag{2}$$

S-ELM is essentially a one-versus-rest classification scheme. For the two-class problem discriminating class k from the remaining ones, the following optimization problem is solved for the calculation of the network output weight vector $\mathbf{w}_k$:

$$\textbf{min:}\quad \mathcal{J}_k = \frac{1}{2}\|\mathbf{w}_k\|_2^2 + c\sum_{i=1}^{N} \xi_{ik}, \tag{3}$$

$$\textbf{s.t.:}\quad t_{ik}\mathbf{w}_k^T\boldsymbol{\phi}_i \geq 1 - \boldsymbol{\xi}_{ik}, \quad i = 1, ..., N, \tag{4}$$

$$\xi_{ik} \geq 0, \quad i = 1, ..., N. \tag{5}$$

The previous optimization problem is solved for all the classes $k = 1, \dots, C$ when the classification problem involves multiple classes, that is, when $C > 2$. By considering the Lagrangian of (3) with respect to the constraints in (4) and (5), and determining its saddle points with respect to $\mathbf{w}_k$ and ξ_{ij}, $\mathcal{J}_k$ is transformed to the following dual quadratic optimization problem:

$$\textbf{min:} \quad \mathcal{L}_{D,k} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{ik}\alpha_{jk}t_{ik}t_{jk}\boldsymbol{\phi}_i^T\boldsymbol{\phi}_j - \sum_{i=1}^{N}\alpha_{ik}, \quad (6)$$

$$\textbf{s.t.:} \quad 0 \leq \alpha_{ik} \leq c, \quad i = 1,...,N. \quad (7)$$

For S-ELMs using random hidden layer parameters, the network output weights $\mathbf{W}_{out} = [\mathbf{w}_1, \ldots, \mathbf{w}_C]$ are obtained by

$$\mathbf{w}_k = \sum_{i=1}^{N}\alpha_{ik}\boldsymbol{\phi}_i, \quad k = 1,\ldots,C \quad (8)$$

and the network output $\mathbf{o}_t$ for a test vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$\mathbf{o}_t = \mathbf{W}_{out}^T\boldsymbol{\phi}_t. \quad (9)$$

In order to exploit kernel formulations, $\mathcal{L}_{D,k}$ in (6) can be written in the form

$$\textbf{min:} \quad \mathcal{L}_{D,k} = \frac{1}{2}(\boldsymbol{\alpha}_k \circ \mathbf{t}_k)^T\mathbf{K}(\boldsymbol{\alpha}_k \circ \mathbf{t}_k) + \mathbf{1}^T\boldsymbol{\alpha}_k, \quad (10)$$

$$\textbf{s.t.:} \quad 0 \leq \alpha_{ik} \leq c, \quad i = 1,...,N, \quad (11)$$

where $\boldsymbol{\alpha}_k \in \mathbb{R}^N$ and $\mathbf{t}_k \in \mathbb{R}^N$ are the Lagrange multipliers and the target vector for the two-class classification problem k, $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones, and $\circ$ denotes the Hadamard (element-wise) product operator. $\mathbf{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$ is the so-called ELM kernel matrix (Huang et al. (2012)).

Therefore, the output $\mathbf{o}_t = [o_{t1}, \ldots, o_{tC}]^T$ of S-ELM for a test vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$o_{tk} = \mathbf{w}_k^T\boldsymbol{\phi}_t = \sum_{i=1}^{N}\alpha_{ik}t_{ik}\boldsymbol{\phi}_i^T\boldsymbol{\phi}_t = (\boldsymbol{\alpha}_k \circ \mathbf{t}_k)^T\mathbf{k}_t, \quad (12)$$

where $\mathbf{k}_t \in \mathbb{R}^N$ is a vector with its elements equal to $k_{ti} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_t$, $i = 1,\ldots,N$. An advantage of calculating the network output vector $\mathbf{o}_t$ through (33) is that, as most of the Lagrange multipliers $\alpha_{ik} = 0$, lesser computations are required, compared with (9).

3. S-ELM exploiting intrinsic graphs

In this section, an optimization scheme for SLFN network training that exploits intrinsic graph structures is described. Similar to S-ELM, we perform a one-versus-rest classification. For the two-class problem discriminating class k from the remaining ones, the following optimization problem is solved for the calculation of the network output weight vector $\mathbf{w}_k$:

$$\textbf{min:} \quad \tilde{\mathcal{J}}_k = \frac{1}{2}\mathbf{w}_k^T\mathbf{S}\mathbf{w}_k + c\sum_{i=1}^{N}\xi_{ik}, \quad (13)$$

$$\textbf{s.t.:} \quad t_{ik}\mathbf{w}_k^T\boldsymbol{\phi}_i \geq 1 - \xi_{ik}, \quad i = 1,...,N, \quad (14)$$

$$\xi_{ik} \geq 0, \quad i = 1,...,N, \quad (15)$$

where $\mathbf{S} \in \mathbb{R}^{L\times L}$ is a matrix expressing geometric relationships of the training data in the ELM space. Following the Graph Embedding approach (Yan et al. (2007)), we assume that the network hidden layer output vectors $\boldsymbol{\phi}_i$ corresponding to the training data $\mathbf{x}_i$, $i = 1,\ldots,N$ are used in order to form the vertex set of a graph $\mathcal{G} = \{\boldsymbol{\Phi}, \mathbf{V}\}$, where $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a similarity matrix whose elements denote geometric relationships between the graph vertices $\boldsymbol{\phi}_i$. $\mathbf{S}$ can be defined by

$$\mathbf{S} = \boldsymbol{\Phi}\mathbf{L}\boldsymbol{\Phi}^T, \quad (16)$$

where $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the graph Laplacian matrix defined by $\mathbf{L} = \mathbf{D} - \mathbf{V}$, with $\mathbf{D}$ being the diagonal degree matrix of $\mathcal{G}$ having elements $D_{ii} = \sum_{j=1}^{N} V_{ij}$. It should be noted here that the calculation of $\mathbf{S}$ in the ELM space $\mathbb{R}^L$, rather than in the input space $\mathbb{R}^D$, has the advantage of nonlinear relationships between the input data $\mathbf{x}_i$ can be better expressed. Using (16), geometric data relationships used in several subspace learning techniques can be exploited. For example, the graph Laplacian matrices used in order to express the total scatter and the within-class scatter of the data in the ELM space are defined as follows:

$$\mathbf{L}_T = \frac{1}{N}\left(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\right), \quad (17)$$

$$\mathbf{L}_w = \mathbf{I} - \sum_{c=1}^{N_c}\frac{1}{N_c}\mathbf{1}_c\mathbf{1}_c^T \quad (18)$$

where $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones, $\mathbf{1}_c \in \mathbb{R}^N$ is a vector with elements $1_{c,i} = 1$ if $l_i = c$ and $1_{c,i} = 0$ if $l_i \neq c$, and N_c denotes the cardinality of class c. In order to exploit class geometric information, graphs describing class data relationships can be used to define appropriate graph weight matrix $\mathbf{V}$ elements as follows:

$$V_{ij}^{(w)} = \begin{cases} \frac{s_{ij}}{N_{c_i}}, & \text{if } l_j = l_i, \\ 0, & \text{otherwise,} \end{cases} \quad (19)$$

where s_{ij} is a measure of similarity between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$, like the heat kernel function:

$$s_{ij} = exp\left(-\frac{\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\|_2^2}{2\sigma^2}\right). \quad (20)$$

σ is a parameter used to scale the Euclidean distance between $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$. Here, we should note that one can exploit different similarity measures in order to exploit a priori information related to the problem at hand. However, the heat kernel is usually used to this end. Finally, kNN graphs describing local class data relationships can be exploited by defining appropriate graph weight matrix $\mathbf{V}$ elements as follows:

$$V_{ij} = \begin{cases} 1, & \text{if } l_i = l_j \text{ and } j \in \mathcal{N}_i, \\ 1, & \text{if } l_i = l_j \text{ and } i \in \mathcal{N}_j, \\ 0, & \text{otherwise,} \end{cases} \quad (21)$$

where $\mathcal{N}_i$ denotes the neighborhood of $\boldsymbol{\phi}_i$. In all our experiments, 5NN graphs have been used.

By taking the Lagrangian of (13) with respect to the constraints in (14) and (15), we obtain

$$\begin{aligned} \tilde{\mathcal{L}}_{D,k} &= \frac{1}{2}\mathbf{w}_k^T\mathbf{S}\mathbf{w}_k + c\sum_{i=1}^{N}\xi_{ik} - \sum_{i=1}^{N}\beta_{ik}\xi_{ik} \\ &- \sum_{i=1}^{N}\alpha_{ik}\left(t_{ik}\mathbf{w}_k^T\boldsymbol{\phi}_i - 1 + \xi_{ik}\right), \end{aligned} \quad (22)$$

where α_{ik} and β_{ik} are the Lagrange multipliers corresponding to the constraints (14) and (15), respectively.

By determining the saddle points of $\tilde{\mathcal{L}}_{D,k}$ with respect to $\mathbf{w}_k$ and ξ_{ij} we obtain

$$\mathbf{w}_k = \mathbf{S}^{-1} \sum_{i=1}^{N} \alpha_{ik} t_{ik} \boldsymbol{\phi}_i \tag{23}$$

and

$$c = \alpha_{ik} + \beta_{ik}. \tag{24}$$

In order to avoid singularity issues of $\mathbf{S}$ ($\mathbf{S}$ will be singular when $L > N$), a regularized version of S is exploited, that is,

$$\tilde{\mathbf{S}} = \mathbf{S} + r\mathbf{I} = \mathbf{\Phi}\mathbf{L}\mathbf{\Phi}^T + r\mathbf{I}, \tag{25}$$

where $r > 0$ is a regularization parameter used in order to exploit the strictly diagonally dominant criterion of nonsingular matrices.

By substituting (23) and (24) in (22), and using (25), $\tilde{\mathcal{J}}_k$ is transformed to the following quadratic optimization problem:

$$\textbf{min:} \quad \tilde{\mathcal{L}}_{D,k} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{ik} \alpha_{jk} t_{ik} t_{jk} \boldsymbol{\phi}_i^T \tilde{\mathbf{S}}^{-1} \boldsymbol{\phi}_j - \sum_{i=1}^{N} \alpha_{ik}, \tag{26}$$

$$\textbf{s.t.:} \quad 0 \leq \alpha_{ik} \leq c, \quad i = 1, ..., N. \tag{27}$$

In the case of random hidden layer parameters, the network output weights $\mathbf{W}_{out} = [\mathbf{w}_1, \ldots, \mathbf{w}_C]$ are obtained by

$$\mathbf{w}_k = \tilde{\mathbf{S}}^{-1} \sum_{i=1}^{N} \alpha_{ik} \boldsymbol{\phi}_i, \quad k = 1, \ldots, C \tag{28}$$

and the network output $\mathbf{o}_t$ for a test vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$\mathbf{o}_t = \mathbf{W}_{out}^T \boldsymbol{\phi}_t. \tag{29}$$

In order to exploit kernel formulations, $\mathbf{w}_k$, $k = 1, \ldots, C$ is expressed as a linear combination of the training data in the ELM space, that is, $\mathbf{w}_k = \mathbf{\Phi}\mathbf{g}_k$, where $\mathbf{g}_k \in \mathbb{R}^N$ is a vector containing the reconstruction weights of $\mathbf{w}_k$ with respect to $\mathbf{\Phi}$. $\tilde{\mathcal{L}}_D$ in (22) becomes

$$\begin{aligned} \tilde{\mathcal{L}}_{D,k} &= \frac{1}{2} \mathbf{g}_k^T (\mathbf{KLK} + r\mathbf{I}) \mathbf{g}_k + c \sum_{i=1}^{N} \xi_{ik} - \sum_{i=1}^{N} \beta_{ik} \xi_{ik} \\ &- \sum_{i=1}^{N} \alpha_{ik} \left(t_{ik} \mathbf{g}_k^T \mathbf{k}_i - 1 + \xi_{ik} \right). \end{aligned} \tag{30}$$

By determining the saddle points of $\tilde{\mathcal{L}}_{D,k}$ with respect to $\mathbf{g}_k$ and ξ_{ik}, $\tilde{\mathcal{L}}_D$ is transformed to the following equivalent dual optimization problem:

$$\textbf{min:} \quad \tilde{\mathcal{L}}_D = \frac{1}{2} (\mathbf{a}_k \circ \mathbf{t}_k)^T \tilde{\mathbf{K}} (\mathbf{a}_k \circ \mathbf{t}_k) + \mathbf{1}^T \mathbf{a}_k, \tag{31}$$

$$\textbf{s.t.:} \quad 0 \leq g_{ik} \leq c, \quad i = 1, ..., N. \tag{32}$$

where $\tilde{\mathbf{K}} = \mathbf{K}(\mathbf{KLK} + r\mathbf{K})^{-1}\mathbf{K}$.

The output $\mathbf{o}_t = [o_{t1}, \ldots, o_{tC}]^T$ of the network for a test vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by

$$\begin{aligned} o_{tk} &= \mathbf{w}_k^T \boldsymbol{\phi}_t = \sum_{i=1}^{N} \alpha_{ik} t_{ik} \boldsymbol{\phi}_i^T \mathbf{\Phi} (\mathbf{KLK} + r\mathbf{K})^{-1} \mathbf{\Phi}^T \boldsymbol{\phi}_t \\ &= (\boldsymbol{\alpha}_k \circ \mathbf{t}_k)^T \mathbf{K} (\mathbf{KLK} + r\mathbf{K})^{-1} \mathbf{k}_t, \end{aligned} \tag{33}$$

where $\mathbf{k}_t \in \mathbb{R}^N$ is a vector with its elements equal to $k_{ti} = \boldsymbol{\phi}_i^T \boldsymbol{\phi}_t$, $i = 1, \ldots, N$.

4. Discussion

By observing (6), (26), and (10), (31), it can be seen that the optimization problems solved by the two approaches in both the original and kernel formulations are similar. In this section, the application of the optimization scheme described in Section 3 in the original ELM space is shown to be equivalent to the application of the S-ELM algorithm (Bai et al. (2014)) in a transformed ELM space, for both the original and kernel S-ELM formulations.

For the optimization scheme exploiting random hidden layer parameters, we work as follows. The singular value decomposition of $\tilde{\mathbf{S}}$ is denoted by

$$\tilde{\mathbf{S}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{34}$$

where $\mathbf{U} \in \mathbb{R}^{L \times L}$ is an orthonormal matrix containing the singular vectors of $\tilde{\mathbf{S}}$ corresponding to its singular values λ_j, $j = 1, \ldots, L$, stored in the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{L \times L}$. Using (34), $\tilde{\mathbf{S}}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T$. The following projection can be defined:

$$\tilde{\boldsymbol{\phi}}_i = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{U}^T \boldsymbol{\phi}_i, \tag{35}$$

where

$$\tilde{\boldsymbol{\phi}}_i^T \tilde{\boldsymbol{\phi}}_j = \boldsymbol{\phi}_i^T \mathbf{U} \mathbf{\Lambda}^{-1} \mathbf{U}^T \boldsymbol{\phi}_j = \boldsymbol{\phi}_i^T \tilde{\mathbf{S}}^{-1} \boldsymbol{\phi}_j. \tag{36}$$

Thus, when random hidden layer parameters are employed, the application of S-ELM (Bai et al. (2014)) to $\tilde{\boldsymbol{\phi}}_i$ is equivalent to the application of the alternative optimization scheme to $\boldsymbol{\phi}_i$.

For the optimization scheme used in the kernel formulation of our approach, we work as follows. By analyzing the matrix $\tilde{\mathbf{K}}$, we obtain

$$\begin{aligned} \tilde{\mathbf{K}} &= \mathbf{K}(\mathbf{KLK} + r\mathbf{K})^{-1}\mathbf{K} \\ &= \mathbf{K}\left[\frac{1}{r}\mathbf{K}^{-1} - \frac{1}{r^2}\left(\mathbf{L}^{-1} + \frac{1}{r}\mathbf{K}\right)^{-1}\right]\mathbf{K} \\ &= \frac{1}{r}\left[\mathbf{I} - \left(\mathbf{L} + r\mathbf{K}^{-1}\right)^{-1}\mathbf{L}\right]\mathbf{K}, \end{aligned} \tag{37}$$

where $\mathbf{K}$ was assumed to be invertible. Thus, when kernel formulations are exploited, the application of S-ELM (Bai et al. (2014)) using the kernel matrix $\tilde{\mathbf{K}} = \frac{1}{r}\left[\mathbf{I} - \left(\mathbf{L} + r\mathbf{K}^{-1}\right)^{-1}\mathbf{L}\right]\mathbf{K}$ is equivalent to the application of our optimization scheme using the kernel matrix $\mathbf{K}$.

It should be noted here that, although this paper focuses on the S-ELM algorithm proposed by Bai et al. (2014), the equivalence of the original ELM space and the transformed ELM space obtained by following the process described earlier is related to the optimization scheme used to calculate the network output weights and not the process used to calculate the network's hidden layer outputs. Thus, the transformed feature space can be exploited in any algorithm that uses a predefined data mapping from the input space to the hidden layer space (e.g., Schmidt et al. (1992); Luo et al. (2014)).

Table 2. Performance (%) on standard classification problems.

Dataset	S-ELM	**S-ELM (18)**	**S-ELM (17)**	**S-ELM (21)**	**S-ELM (19)**
Australian	83.19(±0.39)	**85.29(±0.49)**	**85.31(±0.37)**	**85.41(±0.39)**	**85.25(±0.41)**
Abalone	52.7(±0.01)	**52.85(±0.18)**	52.72(±0.01)	**53.09(±0.26)**	**53.08(±0.14)**
Column	77.87(±0.7)	**78.9(±0.67)**	**79(±0.84)**	**78.74(±0.81)**	**79(±0.47)**
German	71.46(±0.17)	71.51(±0.2)	71.53(±0.18)	**72.02(±0.27)**	71.71(±0.44)
Glass	51.81(±1.14)	**52.43(±1.23)**	**52.61(±1.27)**	**52.09(±1.09)**	**52.1(±1.19)**
Heart	83.18(±1.13)	**83.74(±0.98)**	**83.7(±0.96)**	**83.63(±1.06)**	**83.81(±0.87)**
Indians	74.04(±0.3)	**76.16(±0.33)**	**76.09(±0.31)**	**76.12(±0.32)**	**76.12(±0.35)**
Ionosphere	64.1(±0.01)	**68.24(±1.24)**	64.25(±0.31)	**68.95(±2.4)**	**67.61(±1.02)**
Iris	88.67(±0.89)	**90.8(±1.8)**	**89.73(±1)**	**89.13(±0.77)**	**91.93(±1.9)**
Madelon	59.85(±0.43)	59.97(±0.44)	59.95(±0.39)	**60.02(±0.47)**	59.92(±0.42)
Spect	82.4(±0.53)	**83.48(±0.78)**	82.44(±0.6)	82.55(±0.59)	**83.07(±0.77)**
Spectf	81.31(±0.52)	**82.91(±0.44)**	**81.91(±0.48)**	**81.65(±0.66)**	**82.13(±0.64)**

Table 1. Dataset details.

Dataset	Samples	Dimensions (D)	Classes (C)
Abalone	4177	8	3
Australian	690	14	2
Column	310	6	3
German	1000	24	2
Glass	241	9	6
Heart	270	13	2
Indians	768	8	2
Ionosphere	351	34	2
Iris	150	4	3
Madelon	2600	500	2
Spect	267	22	2
Spectf	267	44	2

5. Experiments

In this section, we present experiments conducted to compare the performance of our method with that of S-ELM. Twelve publicly available datasets were used from the machine learning repository of the University of California Irvine (UCI) (Frank and Asuncion (2010)) to this end. Table 1 provides information concerning the datasets used in our experiments. The datasets were normalized so as to have zero mean and unit standard deviation.

As there is no widely adopted experimental protocol for these datasets, the fivefold cross-validation procedure is performed (Devijver and Kittler (1982)), by taking into account the class labels of the data. That is, the data belonging to each class in five sets were randomly set, and four sets of all classes were used for training and the remaining ones for testing. This process was performed five times, one for each test set in order to complete an experiment. The performance of each algorithm in one experiment was measured by calculating the mean classification rate over all folds. Ten experiments were performed and the performance of each algorithm was measured by calculating the mean classification rate and the observed standard deviation over all experiments.

In all the experiments, we compare the performance of S-ELM (Bai et al. (2014)) with that of our optimization scheme exploiting intrinsic graphs. We employed the RBF kernel function:

$$\mathbf{K}_{RBF}(\mathbf{x}_i, \mathbf{x}_j, \sigma) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right), \tag{38}$$

where the value σ is set equal to the mean Euclidean distance between the training data, that is, the natural scaling factor for each dataset. The optimal value for the regularization parameters c, r has been determined by applying grid search using the values $c = 10^l$, $l = -5, \ldots, 5$ and $r = 10^q$, $q = -5, \ldots, 5$.

Table 2 illustrates the mean classification rates and the observed standard deviation values obtained by applying S-ELM and our S-ELM formulation exploiting the intrinsic graphs described in (17), (18), (19), and (21). We can observe that the exploitation of geometric data information in the S-ELM optimization process can enhance the network's performance in general. By exploiting the data dispersion in the ELM space, S-ELM with the graph described in (17) outperforms S-ELM in most cases. The exploitation of geometric class information in the ELM space also enhances classification performance, as S-ELM exploiting graphs that express intrinsic class geometric information described in (18), (19), and (21) outperform S-ELM in most cases.

6. Conclusions

In this paper, we described an optimization scheme to calculate the output weights of a SLFN network using the S-ELM classification scheme. This optimization scheme exploits data relationships in the ELM space in order to incorporate geometric information in the derived decision function. By following this approach, it is expected that better generalization performance can be achieved, when compared with the solutions obtained by applying the S-ELM algorithm. We have experimentally evaluated the effect of adopting global and local geometric information described in the within-class/total scatter and kNN graphs, respectively. The experimental results confirm our assumptions. The kernel formulations of our method operating in ELM spaces of arbitrary dimensions have also been provided.

We have shown that the application of the optimization scheme that exploits geometric data relationships in the original ELM space is equivalent to the application of the original S-ELM to a transformed ELM space, for both the original and kernel formulations. This fact can be exploited in order to use efficient existing implementations.

7. Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement number 316564 (IMPART).

References

Arvanitidis, G., Tefas, A., 2012. Exploiting graph embedding in support vector machines. IEEE International Workshop on Machine Learning for Signal Processing .

Bai, Z., Huang, G., Wang, W., Wang, H., Westover, M., 2014. Sparse Extreme Learning Machine for classification. IEEE Transactions on Cybernetics 44, 1858–1870.

Bartlett, P., 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Transactions on Information Theory 44, 525–536.

Broomhead, D., Lowe, D., 1988. Multivariable functional interpolation and adaptive networks. Complex Systems 2, 321–355.

Chen, C., 1996. A rapid supervised learning neural network for function interpolation and approximation. IEEE Transactions on Neural Networks 7, 1220–1230.

Devijver, P., Kittler, J., 1982. Pattern Recognition: A Statistical Approach. Prentice-Hall.

Feng, G., Huang, G., Lin, Q., Gay, R., 2009. Error minimized Extreme Learning Machine with growth of hidden nodes and incremental learning. IEEE Transactions on Neural Networks 20, 1352–1357.

Frank, A., Asuncion, A., 2010. Uci machine learning repository.

Huang, G., 2014. An insight into extreme learning machines: Random neurons, random features and kernels. Cognitive Computation 6, 376–390.

Huang, G., Chen, L., 2008. Convex incremental Extreme Learning Machine. Neurocomputing 70, 3056–3062.

Huang, G., Chen, L., Siew, C., 2006. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks 17, 879–892.

Huang, G., Zhou, H., Ding, X., Zhang, R., 2012. Extreme Learning Machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42, 513–529.

Huang, G., Zhu, Q., Siew, C., 2004. Extreme Learning Machine: a new learning scheme of feedforward neural networks. IEEE International Joint Conference on Neural Networks, 2004. Proceedings 2, 985–990.

Iosifidis, A., Tefas, A., Pitas, I., 2013. Minimum Class Variance Extreme Learning Machine for human action recognition. IEEE Transactions on Circuits and Systems for Video Technology 23, 1968–1979.

Iosifidis, A., Tefas, A., Pitas, I., 2014. Regularized Extreme Learning Machine for multi-view semi-supervised action recognition. Neurocomputing 145, 250–262.

Iosifidis, A., Tefas, A., Pitas, I., 2015. On the kernel Extreme Learning Machine classifier. Pattern Recognition Letters 54, 11–17.

Li, M., Huang, G., Saratchandran, P., Sundararajan, N., 2005. Fully complex Extreme Learning Machine. Neurocomputing 68, 306–314.

Liang, N., Huang, G., Saratchandran, P., Sundararajan, N., 2006. A fast and accurate on-line sequential learning algorithm for feedforward networks. IEEE Transactions on Neural Networks 17, 1411–1423.

Liu, X., Lin, S., Fang, J., Xu, Z., 2015a. Is Extreme Learning Machine feasible? a theoretical assessment (Part I). IEEE Transactions on Neural Networks and Learning Systems 26, 7–20.

Liu, X., Lin, S., Fang, J., Xu, Z., 2015b. Is Extreme Learning Machine feasible? a theoretical assessment (Part II). IEEE Transactions on Neural Networks and Learning Systems 26, 21–34.

Luo, J., Vong, C., Wong, P., 2014. Sparse Bayesian Extreme Learning Machine for multi-classification. IEEE Transactions on Neural Networks and Learning Systems 25, 836–843.

Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A., 2010. OP-ELM: Optimally Pruned Extreme Learning Machine. IEEE Transactions on Neural Networks 21, 158–162.

Orphanidis, G., Tefas, A., 2012. Exploiting subclass information in support vector machines. International Conference on Pattern Recognition .

Pao, Y., Park, G., Sobajic, D., 1994. Learning and generalization characteristics of random vector functional-link net. Neurocomputing 6, 163–180.

Rahimi, A., Recht, B., 2008. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. Advances in Neural Information Processing Systems .

Schmidt, W., Kraaijveld, M., Duin, R., 1992. Feedforward neural networks with random weights. International Conference on Pattern Recognition .

Sha, F., Lin, Y., Saul, L., Lee, D., 2007. Multiplicative updates for nonnegative quadratic programming. Neural Computation 19, 2004–2031.

Wang, Y., Cao, F., Yuan, Y., 2011. A study on effectiveness of Extreme Learning Machine. Neurocomputing 74, 2483–2490.

Widrow, B., Greenblatt, A., Kim, Y., Park, D., 2013. The no-prop algorithm: A new learning algorithm for multilayer neural networks. Neural Networks 37, 182–188.

Yan, S., Xu, D., Zhang, B., Zhang, H., 2007. Graph Embedding and extensions: A general framework for dimensionality reduction. IEEE Transactions on Pattern Analysis and Machine Intelligence 29, 40–51.