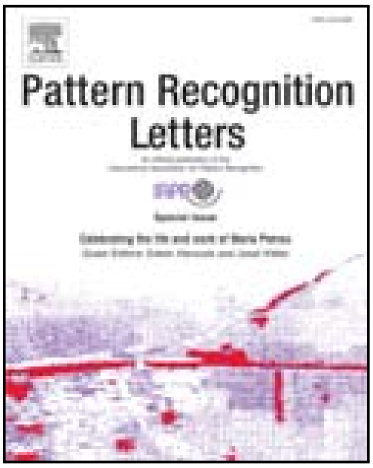

On the Kernel Extreme Learning Machine Classifier

Alexandros Iosifidis, Anastastios Tefas, Ioannis Pitas

PII: S0167-8655(14)00370-5
DOI: 10.1016/j.patrec.2014.12.003
Reference: PATREC 6140

To appear in: *Pattern Recognition Letters*

Received date: 10 September 2014
Accepted date: 1 December 2014

Pattern Recognition Letters
journal homepage: www.elsevier.com

On the Kernel Extreme Learning Machine Classifier

Alexandros Iosifidis[a,**], Anastastios Tefas[a], Ioannis Pitas[a]

[a] *Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece*

ARTICLE INFO

Article history:

2000 MSC:
41A05
41A10
65D05
65D17

Keywords:
Extreme Learning Machine
Single-hidden Layer Networks
Infinite Networks

ABSTRACT

In this paper, we discuss the connection of the kernel versions of the ELM classifier with infinite Single-hidden Layer Feedforward Neural networks and show that the original ELM kernel definition can be adopted for the calculation of the ELM kernel matrix for two of the most common activation functions, i.e., the RBF and the sigmoid functions. In addition, we show that a low-rank decomposition of the kernel matrix defined on the input training data can be exploited in order to determine an appropriate ELM space for input data mapping. The ELM space determined from this process can be subsequently used for network training using the original ELM formulation. Experimental results denote that the adoption of the low-rank decomposition-based ELM space determination leads to enhanced performance, when compared to the standard choice, i.e., random input weights generation.

1. Introduction

Extreme Learning Machine (ELM) is an algorithm for Single-hidden Layer Feedforward Neural (SLFN) networks training (Huang et al. (2004, 2006)) that leads to fast network training requiring low human supervision. The main idea in ELM is that the network hidden layer parameters need not to be learned, but can be randomly assigned. The network output parameters can be, subsequently, analytically calculated. Despite the fact that the determination of the network hidden layer outputs is based on randomly assigned input weights, it has been proven that SLFN networks trained by using the ELM algorithm have the properties of global approximators (Huang et al. (2006); Zhang et al. (2012)). In the original ELM algorithm (Huang et al. (2004)), the trained network not only tends to reach the smallest training error, but also the smallest output weight norm, which indicates good generalization performance (Bartlett (1998)). In addition, several optimization schemes have been proposed in the literature for the calculation of 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)), while it has been recently shown that ELM networks are able to outperform other state-of-the-art classifiers, like Support Vector Machine (SVM) (Huang et al. (2012); Huang (2013)). Due to its effectiveness and its fast learning process, the ELM network has been adopted in many classification problems (Viangteeravat et al. (2007); Rong et al. (2008); Lan et al. (2008); Helmy and Rasheed (2009); Deng et al. (2004); Minhas et al. (2010); Miche et al. (2011); Suresh et al. (2010)).

As has been pointed out in (Iosifidis et al. (2013)), the ELM algorithm can be considered to be a learning process formed by two processing steps. The first step corresponds to a (usually nonlinear) mapping process of the input space $\mathbb{R}^D$ to a (usually) high-dimensional feature space $\mathbb{R}^L$ (noted as ELM space), preserving some properties of interest for the training data. In the second step, an optimization scheme is employed for the determination of a linear projection of the high-dimensional data to a low-dimensional feature space $\mathbb{R}^C$, where classification is performed by a linear classifier. In the above-described process, the dimensionality L of the ELM space is usually empirically chosen. In order to find the optimal ELM space dimensionality several methods have been proposed (Miche et al. (2010);

[**] Corresponding author: Tel,Fax: +30-2310996304;
e-mail: aiosif@aiia.csd.auth.gr (Alexandros Iosifidis)

Huang and Chen (2008)). Such methods either start by using a large number of hidden neurons and iteratively decrease it as long as the classification residual error remains above a predefined threshold, or start by using a small number of hidden neurons and iteratively increase it in order to achieve an adequate training performance.

In order to avoid the application of time-consuming algorithms for the determination of the ELM space dimensionality, kernel versions of the ELM classifier have been recently proposed (Huang et al. (2012); Bai et al. (2014)). The idea in these ELM variants is that the network hidden layer outputs need not to be calculated, but they can be inherently encoded in the so-called ELM kernel matrix defined by $\mathbf{K} = \mathbf{\Phi}^T \mathbf{\Phi}$, where $\mathbf{\Phi} \in \mathbb{R}^{L \times N}$ refers to the training data representations in the ELM space and N is the number of training data. That is, the ELM kernel matrix is defined on the network hidden layer outputs $\phi_i,\ i = 1, \dots, N$ and not on the original D-dimensional training data $\mathbf{x}_i$. This contradicts with the common strategy followed in the literature that adopts the standard kernel approach for the calculation of $\mathbf{K}$ (Huang et al. (2012); Bai et al. (2014)), since in the standard kernel approach the corresponding kernel matrix is a function of the input data $\mathbf{x}_i$. In order to avoid this contradiction, the Cholesky decomposition of the kernel matrix defined on the input training data $\mathbf{x}_i$ has been employed in (Parviainen et al. (2010)), in order to calculate an appropriate matrix $\mathbf{\Phi} \in \mathbb{R}^{N \times N}$. While this process leads to correctly defined network hidden layer outputs for the training data, it has the following drawbacks: 1) the use of Cholesky decomposition sets the restriction that the dimensionality of the obtained ELM space must be equal to N and 2) since the obtained matrix $\mathbf{\Phi}$ is a lower triangular matrix, each training sample is (actually) mapped to an ELM space of different dimensions.

In this paper, we show that for two types of network hidden layer activation functions, the original ELM matrix definition can be exploited for ELM networks training. To this end, we discuss the connection of the kernel ELM networks to infinite SLFN networks (Neal (1996); Williams (1998)). In addition, we show that a low-rank decomposition of the kernel matrix defined on the input training data can be employed for the determination of an appropriate ELM space that overcomes the drawbacks of the Cholesky decomposition used in (Parviainen et al. (2010)). Finally, we experimentally compare the performance of ELM networks exploiting randomly assigned hidden layer parameters with the performance of ELM networks trained by using a low-rank decomposition of the kernel matrix for the determination of the hidden layer outputs. Experimental results show that, for the same ELM space dimensionality, the latter choice leads to enhance classification performance.

The rest of the paper is structured as follows. Section 2 provides an overview of the ELM algorithm. In Section 5, we discuss the connection ELMs with infinite networks. In Section 4, we show that a low-rank decomposition of the kernel matrix defined on the input training data can be employed for the determination of an appropriate ELM space. Experiments conducted in real datasets are provided in Section 6. Finally, conclusions are drawn in Section 7.

2. Overview of ELM networks

Let us denote by $\{\mathbf{x}_i, l_i\}_{i=1,\dots,N}$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding class labels $l_i \in \{1, \dots, C\}$ that can be used 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 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, i.e., when $l_i = k$, and to $t_{ik} = -1$, otherwise. 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 $\mathbf{W}_{in}$ and $\mathbf{w}_k$ is the k-th column of $\mathbf{W}_{out}$. 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 $\mathbf{\Phi} = [\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N]$, the network response for all the training data $\mathbf{O} \in \mathbb{R}^{C \times N}$ can be expressed in a matrix form as:

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

The original ELM algorithm (Huang et al. (2004)) assumes zero training error. That is, it is assumed that $\mathbf{o}_i = \mathbf{t}_i,\ i = 1, \dots, N$, or by using a matrix notation $\mathbf{O} = \mathbf{T}$, where $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]$ is a matrix containing the network target vectors. By using (2), the network output weights $\mathbf{W}_{out}$ can be analytically calculated by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^{-1} \mathbf{\Phi}\, \mathbf{T}^T = \mathbf{\Phi}^\dagger\, \mathbf{T}^T. \tag{3}$$

In the case where $L > N$, the calculation of the network output weights $\mathbf{W}_{out}$ through (3) is inaccurate, since the matrix $\mathbf{\Phi}\mathbf{\Phi}^T$ is singular. 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 (Huang et al. (2012)). In this case, the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{N} \|\boldsymbol{\xi}_i\|_2^2, \tag{4}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \qquad i = 1, ..., N, \tag{5}$$

where $\boldsymbol{\xi}_i \in \mathbb{R}^C$ is the error vector corresponding to $\mathbf{x}_i$ and λ is a parameter denoting the importance of the training error in the optimization problem, satisfying $\lambda > 0$. By substituting the constraints (5) in (4) and determining the saddle point of $\mathcal{J}_{RELM}$ with respect to $\mathbf{W}_{out}$, the network output weights are obtained by:

$$\mathbf{W}_{out} = \left(\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{\lambda}\mathbf{I}\right)^{-1} \mathbf{\Phi}\mathbf{T}^T, \tag{6}$$

or

$$\mathbf{W}_{out} = \mathbf{\Phi}\left(\mathbf{\Phi}^T\mathbf{\Phi} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{T}^T = \mathbf{\Phi}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{T}^T, \qquad (7)$$

where $\mathbf{I} \in \mathbb{R}^{L\times L}$ is the identity matrix.

In the latter case, after the calculation of the network output weights $\mathbf{W}_{out}$, the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T\boldsymbol{\phi}_l = \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{\Phi}^T\boldsymbol{\phi}_l = \mathbf{T}\left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}\mathbf{k}_l, \qquad (8)$$

where $\mathbf{k}_l$ is the ELM kernel vector for $\mathbf{x}_l$.

Recently, an optimization scheme leading to sparse ELM solution has been proposed in (Bai et al. (2014)). This ELM variant solves the following optimization problem for the calculation of the the network output weights:

$$\textbf{Minimize:} \quad \mathcal{J}_{S-ELM} = \frac{1}{2}\|\mathbf{w}_k\|_2^2 + \frac{c}{2}\sum_{i=1}^{N}\xi_{ik}, \qquad (9)$$

$$\textbf{Subject to:} \quad t_{ik}\mathbf{w}_k^T\boldsymbol{\phi}_i \geq 1 - \boldsymbol{\xi}_{ik}, \quad i = 1,...,N, \qquad (10)$$

$$\xi_{ik} \geq 0, \quad i = 1,...,N. \qquad (11)$$

The above optimization problem is solved for all the classes $k = 1,\dots,C$ in an One-Versus-Rest manner for the calculation of $\mathbf{W}_{out}$, in the case of multiple classes. By taking the Lagrangian of (9) with respect to the constraints in (10) and (11), and determining its saddle point, $\mathcal{J}_{S-ELM}$ is transformed to the following dual quadratic optimization problem:

$$\textbf{Minimize:} \quad \mathcal{L}_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_{ik}\alpha_{jk}t_{ik}t_{jk}\boldsymbol{\phi}_i^T\boldsymbol{\phi}_j - \sum_{i=1}^{N}\alpha_i, (12)$$

$$\textbf{Subject to:} \quad 0 \leq \alpha_{ik} \leq c, \quad i = 1,...,N. \qquad (13)$$

Therefore the output $\mathbf{o}_l = [o_{l1},\dots,o_{lC}]^T$ of the sparse ELM for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$o_{lk} = \mathbf{w}_k^T\boldsymbol{\phi}_l = \sum_{i=1}^{N}\alpha_{ik}t_{ik}\boldsymbol{\phi}_i^T\boldsymbol{\phi}_l = \sum_{i=1}^{N}\alpha_{ik}t_{ik}\mathbf{k}_l, \quad k = 1,\dots,C. \qquad (14)$$

An advantage of the solution in (14), when compared to the one in (8), is that (usually) most of the values in α_{ik} are equal to zero, thus, leading to faster computation of $\mathbf{o}_l$.

3. Connection of ELM to infinite SLFNs

As has been described above, in ELMs the network outputs can be obtained by exploiting only dot products of the data representations in the ELM space, as detailed in (8) and (14) for the ELM and Sparse ELM cases, respectively. By expressing such dot products using the ELM kernel matrix $\mathbf{K}$, the number of hidden layer neurons needs not to be determined. That is, the network hidden layer may consist of arbitrary (even infinite) number of neurons. As has been shown in (Parviainen et al. (2010)), an ELM network consisting of a sufficiently large number of hidden layer neurons operates as an approximation of an infinite SLFN network (Neal (1996); Williams (1998)).

In (Neal (1996); Williams (1998)), it has been proven that infinite SLFN networks employing a linear activation function for the network output layer (which is the case of ELMs) can be modeled as Gaussian processes. Specifically, by letting the number of hidden layer neurons go to infinity and setting a Gaussian prior to the hidden layer weights $\mathbf{v}_k$, $k = 1,...,L$, $L \to \infty$, the evaluation of $E_\mathbf{v}[\boldsymbol{\phi}_i, \boldsymbol{\phi}_j]$ for all i, j in the training and test sets leads to the determination of the covariance function needed to describe the SLFN network as a Gaussian process. These expectations are obtained by integrating over the relevant probability distributions of the biases and the input weights $\mathbf{v}$.

For a Gaussian prior over the distribution of $\mathbf{v}_k$ so that $\mathbf{v}_k \sim N(0, \sigma_v^2\mathbf{I})$, the adoption of an RBF hidden layer activation function $\phi(\mathbf{x}_i, \mathbf{v}_k) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{v}_k\|_2^2}{2\sigma_g^2}\right)$ leads to a covariance function of the form:

$$\mathbf{C}(\mathbf{x}_i, \mathbf{x}_j) = \left(\frac{\sigma_e}{\sigma_v}\right)^D exp\left(-\frac{\|\mathbf{x}_i\|_2^2}{2\sigma_m^2}\right) exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_s^2}\right) exp\left(-\frac{\|\mathbf{x}_j\|_2^2}{2\sigma_m^2}\right), \qquad (15)$$

where $\sigma_e^2 = (\sigma_v^2 + \sigma_g^2)/(\sigma_v^2\sigma_g^2)$, $\sigma_s^2 = 2\sigma_v^2 + \sigma_g^4/\sigma_v^2$ and $\sigma_m^2 = 2\sigma_v^2 + \sigma_g^2$. If $\sigma_v^2 \to \infty$, we find that $\mathbf{C}(\mathbf{x}_i, \mathbf{x}_j) \propto exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_s^2}\right)$, i.e. the RBF kernel function defined on the training (and test) data $\mathbf{x}_i$.

For the case of sigmoid hidden layer activation function, by making the assumption that $\mathbf{v}_k$, $k = 1,\dots,L$ are drawn from a zero-mean Gaussian distribution with covariance matrix $\mathbf{\Sigma}$, i.e., $\mathbf{v}_k \sim N(0, \sigma_v^2\mathbf{\Sigma})$, the corresponding covariance function is given by (Williams (1998)):

$$\mathbf{C}(\mathbf{x}_i, \mathbf{x}_j) = \frac{2}{\pi}sin^{-1}\frac{\tilde{\mathbf{x}}_i^T\mathbf{\Sigma}\tilde{\mathbf{x}}_j}{\sqrt{\left(1 + \tilde{\mathbf{x}}_i^T\mathbf{\Sigma}\tilde{\mathbf{x}}_i\right)\left(1 + \tilde{\mathbf{x}}_j^T\mathbf{\Sigma}\tilde{\mathbf{x}}_j\right)}}, \qquad (16)$$

where $\tilde{\mathbf{x}}_i$ is the augmented input vector $\tilde{\mathbf{x}}_i = [1, \mathbf{x}_i^T]^T$.

From the above, it can be seen that by adopting the covariance functions determined for the RBF or the sigmoid hidden layer activation functions for the determination of $\mathbf{K}$, $\mathbf{k}_l$, ELM networks are approximations of infinite SLFN networks. Thus, it can be seen that the adoption of the RBF kernel function (defined over the input data $\mathbf{x}_i$) corresponds to the case of RBF hidden layer activation function under the assumption of Gaussian distribution for the randomly sampled input weights $\mathbf{W}_{in}$ $N(0, \sigma_v^2\mathbf{I})$ using $\sigma_v^2 \to \infty$. It should be noted though that, the adoption of the sigmoid kernel for the calculation of the ELM kernel matrix $\mathbf{K}$ (defined over the input data $\mathbf{x}_i$) does not correspond to the case of sigmoid hidden layer activation function (in this case the covariance function in (16) should be used). This is also the case for most of the kernel functions defined on the input data $\mathbf{x}_i$, where appropriate covariance functions should be defined and used. However, as will be discussed in the next Section, appropriate ELM spaces can be obtained by employing a low-rank decomposition of the standard kernel matrix $\mathbf{K}$ defined on the input data.

4. ELM space determination based on low-rank decomposition of K

By exploiting the fact that the ELM kernel is defined by $\mathbf{K} = \mathbf{\Phi}^T \mathbf{\Phi}$ and that the maximal dimensionality of the manifold where the training data belong to is equal to N, a low-rank decomposition of the kernel matrix defined on the input training data can be employed for the determination of an appropriate ELM space. Let us denote by $\mathbf{U}$, $\mathbf{V} \in \mathbb{R}^{N \times N}$ two orthogonal matrices and $\mathbf{S} \in \mathbb{R}^{N \times N}$ a diagonal matrix obtained by applying Singular Value Decomposition (SVD) on $\mathbf{K}$, i.e.:

$$\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{17}$$

where we assume that the singular values appearing in $\mathbf{S}$ are sorted in descending order. $\mathbf{U}$, $\mathbf{V}$ are sorted accordingly. Since $\mathbf{K}$ is symmetric and positive semi-definite, $\mathbf{U} = \mathbf{V}$ and, thus, $\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{U}^T$. We can define the hidden layer outputs for the training data $\mathbf{x}_i,\ i = 1, \dots, N$ to be equal to $\mathbf{\Phi} = \mathbf{S}^{\frac{1}{2}}\mathbf{U}^T$.

In the case where we assume that the network hidden layer consists of $r < N$ neurons, we can keep only r of the leading singular values (and the corresponding singular vectors), in order to determine a low-rank approximation of $\mathbf{K}$, i.e.:

$$\tilde{\mathbf{K}} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{U}}^T, \tag{18}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{N \times r}$ and $\tilde{\mathbf{S}} \in \mathbb{R}^{r \times r}$. In this case, $\tilde{\mathbf{\Phi}} = \tilde{\mathbf{S}}^{\frac{1}{2}}\tilde{\mathbf{U}}^T$, leading to the determination of an ELM space having dimensionality equal to $r < N$. In both cases, a test sample $\mathbf{x}_l$ can be mapped to the previously determined ELM space by applying:

$$\boldsymbol{\phi}_l = \tilde{\mathbf{\Phi}}^{\dagger}\mathbf{k}_l \tag{19}$$

and the network response can be obtained by using (8), (14) for the ELM and S-ELM algorithms, respectively. It should be noted here that, similar low-rank approximations have been found to be effective in other classification schemes where they have been used for regularization (Gu and Guo (2012); Smola et al. (2000); Pekalska and Haasdonk (2009)).

5. Time complexity analysis

In the following, we provide a time complexity analysis for networks trained using the ELM algorithm exploiting random hidden weights, the KELM and the proposed method exploiting low-rank approximation of the kernel ELM matrix.

The ELM algorithm exploiting random hidden weights requires the following processing steps:

- Calculation of the hidden layer output matrix $\mathbf{\Phi}$ having time complexity equal to $O(NLD)$.
- Calculation of the network output weight matrix $\mathbf{W}_{out}$ through (6), having time complexity equal to $O(L^3 + L^2N + LNC)$.

The KELM algorithm requires the following processing steps:

- Calculation of the kernel matrix $\mathbf{K}$ having time complexity equal to $O(N^2D)$.
- Calculation of the network output weight matrix $\mathbf{W}_{out}$ through (7) having time complexity equal to $O(2N^3 + CN^2)$.

Finally, the proposed method requires the following processing steps:

- Calculation of the kernel matrix $\mathbf{K}$ having time complexity equal to $O(N^2D)$.
- Calculation of the SVD approximation of $\mathbf{K}$ having time complexity equal to $O(N^3)$.
- Solution of the problem in (6) having time complexity equal to $O(r^3 + r^2N + rNC)$.

From the above, the time complexity of KELM is equal to $O(2N^3 + (C + D)N^2)$, while the time complexity of the proposed method is equal to $O(N^3 + DN^2 + r^3 + r^2N + rNC)$. As will be shown in the experimental section, the proposed method achieves satisfactory performance for values $r << N$ and, thus, the terms involving r in its time complexity are not significant, when compared to the terms involving N^3. Thus, we can conclude that the computational complexity of the proposed method is the same with that of the KELM algorithm, i.e. $O(N^3)$.

The time complexity of the ELM exploiting random hidden weights is equal to $O(L^3 + L^2N + (C + D)LN) \simeq O(L^3 + L^2N)$. In the case where the number of hidden layer neurons is selected to be much lower than the number of training data, i.e. $L << N$, the time complexity of KELM and of the proposed method is higher than the one of the ELM exploiting random hidden weights. However, as will be seen in the experimental section, in order to achieve performance comparable to that of the KELM and the proposed method, the ELM exploiting random hidden weights requires a high number of hidden layer neurons ($L \propto N$). In that case, its time complexity is comparable with that of KELM and the proposed method.

6. Experiments

In this Section, we present experiments conducted in order to illustrate that the adoption of a low-rank decomposition of the kernel matrix for the determination of the hidden layer outputs (generally) outperforms the random assignment choice.

We have employed twelve publicly available datasets to this end: six from the machine learning repository of University of California Irvine (UCI) (Frank and Asuncion (2010)) and six facial image datasets, namely the AR, ORL and Extended YALE-B (designed for face recognition) and the BU, COHN-KANADE and JAFFE (designed for facial expression recognition) datasets. Table 1 provides information concerning the UCI data sets used, while a brief description of the facial image datasets is provided in the following subsections. Experimental results are provided in subsection 6.3.

In all the experiments we apply the regularized ELM (6) and the sparse ELM (14) algorithms for different ELM space dimensionalities. In the case of random hidden layer neuron

Table 1. UCI data sets details.

Data set	Samples	Dimensions	Classes
Libras	360	90	15
Madelon	2600	500	2
Opt. Digits	5620	64	10
Segmentation	2310	19	7
Synth. Control	600	60	6
Tic-tac-toe	958	9	2

parameter assignment, we have employed the RBF activation function:

$$\Phi_{RBF}(\mathbf{x}_i, \mathbf{v}_j, \sigma) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{v}_j\|_2^2}{2\sigma^2}\Big), \tag{20}$$

where the value σ is set equal to the mean Euclidean distance between the training data $\mathbf{x}_i$ and the network input weights $\mathbf{v}_j$, which is the natural scaling factor for the Euclidean distances between $\mathbf{x}_i$ and $\mathbf{v}_j$. In the case of low-rank decomposition-based ELM space determination, we employed the RBF kernel function:

$$\mathbf{K}_{RBF}(\mathbf{x}_i, \mathbf{x}_j, \sigma) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big), \tag{21}$$

where the value σ is set equal to the mean Euclidean distance between the training data. The optimal value for the regularization parameter λ has been determined by applying linear search using the values $\lambda = 10^r,\ r = -3, \dots, 3$.

6.1. Face recognition datasets

The AR dataset (Martinez and Kak (2001)) consists of over 4000 facial images depicting 70 male and 56 female faces. In our experiments, we have used the preprocessed (cropped) facial images provided by the database, depicting 100 persons (50 males and 50 females) having a frontal facial pose, performing several expressions (anger, smiling and screaming), in different illumination conditions (left and/or right light) and with some occlusions (sun glasses and scarf). Each person was recorded in two sessions, separated by an interval of two weeks. Example images of the dataset are illustrated in Figure 1.

The ORL dataset (Samaria and Harter (1994)) consists of 400 facial images depicting 40 persons (10 images each). The images were captured at different times and with different conditions, in terms of lighting, facial expressions (smiling/not smiling) and facial details (open/closed eyes, with/without glasses). Facial images were taken in frontal position with a tolerance for face rotation and tilting of up to 20 degrees. Example images of the dataset are illustrated in Figure 2.

The Extended YALE-B dataset (Lee et al. (2005)) consists of facial images depicting 38 persons in 9 poses, under 64 illumination conditions. In our experiments, we have used the frontal cropped images provided by the database. Example images of the dataset are illustrated in Figure 3.

6.2. Facial expression recognition datasets

The BU dataset (Yin et al. (2006)) consists of facial images depicting over 100 persons (60% feamale and 40% male)

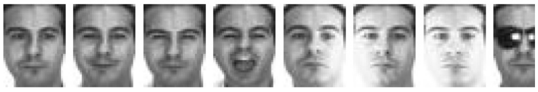

Fig. 1. *Facial images depicting a person of the AR dataset.*

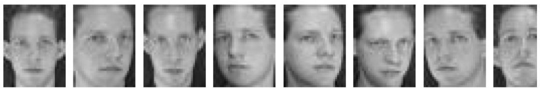

Fig. 2. *Facial images depicting a person of the ORL dataset.*

with a variety of ethnic/racial background, including White, Black, East-Asian, Middle-East Asian, Hispanic Latino and other types of persons. All expressions, except the neutral one, are expressed at four intensity levels. In our experiments, we

Table 2. Experimental results on UCI datasets.

		ELM		S-ELM	
	L	Random	Low-rank	Random	Low-rank
Libras	50	66.8 (±1.69)	**76.77 (±1.68)**	76.88 (±1.72)	**85.66 (±0.52)**
	100	70.5 (±1.54)	**82.49 (±0.91)**	78.38 (±1.33)	**86.17 (±0.82)**
	250	75.2 (±1.9)	**86.34 (±0.54)**	78.93 (±1.41)	**86.49 (±0.64)**
	500	77.21 (±1.6)	**86.32 (±0.52)**	79.24 (±1.64)	**86.52 (±0.69)**
	1000	78.4 (±1.08)	-	79.35 (±1.35)	-
Madelon	50	53.87 (±0.94)	**60.34 (±0.33)**	53.39 (±1.02)	**60.32 (±0.29)**
	100	55.4 (±0.97)	**60.04 (±0.35)**	54.75 (±0.86)	**59.79 (±0.43)**
	250	56.96 (±0.82)	**59.85 (±0.38)**	56.11 (±0.92)	**58.83 (±0.44)**
	500	57.87 (±0.53)	**59.43 (±0.47)**	56.61 (±0.31)	**58.44 (±0.85)**
	1000	58.54 (±0.44)	**59.45 (±0.43)**	57.98 (±0.48)	**58.33 (±0.74)**
Opt. Digits	50	92.02 (±0.27)	**98.21** (±0.04)	**96.92 (±0.27)**	94.12 (±0.09)
	100	94.51 (±0.23)	**98.5** (±0.13)	**97.34 (±0.23)**	96.51 (±0.09)
	250	97.16 (±0.19)	**98.75** (±0.1)	97.45 (±0.19)	**98.42 (±0.09)**
	500	98.08 (±0.15)	**98.8** (±0.09)	97.68 (±0.15)	**98.81 (±0.07)**
	1000	98.54 (±0.13)	**98.83** (±0.1)	97.89 (±0.13)	**99.01 (±0.06)**
Segmentation	50	90.19 (±0.42)	**96.47 (±0.14)**	95.85 (±0.23)	**96.47 (±0.21)**
	100	91.79 (±0.3)	**96.63 (±0.16)**	95.9 (±0.28)	**96.63 (±0.21)**
	250	93.09 (±0.13)	**96.67 (±0.22)**	96.2 (±0.2)	**96.67 (±0.12)**
	500	93.55 (±0.12)	**96.68 (±0.2)**	96.39 (±0.24)	**96.68 (±0.14)**
	1000	93.77 (±0.15)	**96.68 (±0.19)**	96.52 (±0.18)	**96.68 (±0.14)**
Synth. Control	50	84.3 (±1.34)	**97.78 (±0.45)**	97.05 (±0.96)	**98.92** (±0.24)
	100	92.7 (±1.23)	**97.43 (±0.39)**	97.63 (±0.8)	**98.9** (±0.2)
	250	96.03 (±0.61)	**97.55 (±0.36)**	98.08 (±0.48)	**98.97 (±0.21)**
	500	96.27 (±0.34)	**97.55 (±0.33)**	98.1 (±0.29)	**98.97 (±0.21)**
	1000	96.33 (±0.53)	-	98.28 (±0.26)	-
Tic-tac-toe	50	80.05 (±2.09)	**82.24 (±0.59)**	**90.85 (±5.22)**	82.24 (±0.59)
	100	**88.11 (±1.34)**	87.96 (±0.71)	**93.12 (±4.07)**	87.96 (±0.71)
	250	96.49 (±0.69)	**98.81 (±0.28)**	93.45 (±2.42)	**98.81 (±0.28)**
	500	98.35 (±0.01)	**98.81 (±0.32)**	97.93 (±0.19)	**98.81 (±0.32)**
	1000	98.33 (±0.01)	**98.82 (±0.36)**	98.33 (±0.01)	**98.82 (±0.36)**

Table 3. Experimental results on the face recognition datasets.

		ELM		S-ELM	
	L	Random	Low-rank	Random	Low-rank
AR	50	**73.45 (±1.19)**	64.99 (±0.82)	48.64 (±2.58)	**84.23 (±0.48)**
	100	**90.11 (±0.69)**	85.44 (±0.42)	77.32 (±1.19)	**90.28 (±0.41)**
	250	**97.05 (±0.29)**	96.79 (±0.24)	**93.95 (±0.37)**	93.21 (±0.38)
	500	98.36 (±0.17)	**98.92 (±0.16)**	**96.53 (±0.39)**	93.94 (±0.38)
	1000	98.77 (±0.13)	**99.31 (±0.12)**	**97.03 (±0.25)**	94.32 (±0.34)
ORL	50	88.83 (±1.14)	**95.37 (±0.65)**	58.1 (±2.28)	**97.5 (±0.54)**
	100	93.8 (±1.19)	**97.97 (±0.3)**	75.88 (±1.42)	**97.58 (±0.49)**
	250	95.63 (±0.8)	**98.23 (±0.34)**	84 (±0.93)	**97.65 (±0.49)**
	500	96.38 (±0.59)	**98.27 (±0.36)**	84.95 (±1.38)	**97.65 (±0.47)**
	1000	96.5 (±0.53)	-	85.28 (±1.2)	-
YALE	50	**69.42 (±1.09)**	69.2 (±0.59)	73.27 (±1.44)	**88.5 (±0.18)**
	100	82.24 (±0.73)	**83.59 (±0.48)**	86.11 (±0.59)	**91.06 (±0.26)**
	250	91.73 (±0.41)	**94.57 (±0.32)**	92.27 (±0.28)	**92.75 (±0.29)**
	500	95.51 (±0.28)	**97.59 (±0.15)**	93.09 (±0.44)	**93.35 (±0.26)**
	1000	97.04 (±0.15)	**98.28 (±0.1)**	93.52 (±0.36)	**93.56 (±0.31)**

Fig. 3. *Facial images depicting a person of the Extended YALE-B dataset.*

have employed the images depicting the most expressive intensity of each facial expression. Example images of the dataset are illustrated in Figure 4.

Fig. 4. *Facial images depicting a person of the BU dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

The COHN-KANADE dataset (Kanade et al. (2000)) consists of facial images depicting 210 persons of age between 18 and 50 (69% female, 31% male, 81% Euro-American, 13% Afro-American and 6% other groups). We have randomly selected 35 images for each facial expression, i.e., anger, disgust, fear, happyness, sadness, surprise and neutral ones. Example images of the dataset are illustrated in Figure 5.

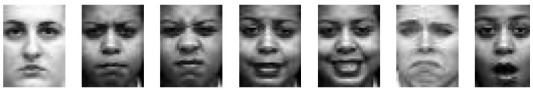

Fig. 5. *Facial images from the COHN-KANADE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

The JAFFE dataset (Lyons et al. (1998)) consists of 210 facial images depicting 10 Japanese female persons. Each expression is depicted in 3 images for each person. Example images of the dataset are illustrated in Figure 6.

6.3. Experimental Results

In our first set of experiments, we have applied the algorithms on the UCI datasets. Since there is no widely adopted experimental protocol for these datasets, we perform the five-fold cross-validation procedure (Devijver and Kittler (1982)), by taking into account the class labels of the data. That is, we randomly split the data belonging to each class in five sets and we use four sets of all classes for training and the remaining ones for testing. This process is performed five times, one for each test set in order to complete an experiment. The performance of each algorithm in one experiment is measured by calculating the mean classification rate over all folds. We perform 10 experiments and measure the performance of each algorithm by calculating the mean classification rate and the observed standard deviation over all experiments.

Table 2 illustrates the performance of the ELM and S-ELM algorithms for different hidden layer dimensionalities L for the

Table 4. Experimental results on the facial expression recognition datasets.

		ELM		S-ELM	
	L	Random	Low-rank	Random	Low-rank
BU	50	48.91 (±1.88)	**57.06 (±1.11)**	44.87 (±1.5	**56.17 (±0.9)**
	100	55.09 (±1.27)	**58.99 (±1.08)**	50.97 (±1.47)	**57.71 (±1.05)**
	250	60.93 (±1.02)	**60.99 (±1.06)**	57.8 (±1.94)	**59.09 (±0.54)**
	500	**62.6 (±1.12)**	61.73 (±0.79)	**61.17 (±1.25)**	59.61 (±0.94)
	1000	**62.91 (±1.21)**	62.1 (±0.79)	**61.61 (±1.17)**	59.74 (±0.95)
KANADE	50	54.98 (±2.18)	**59.76 (±2.74)**	42.33 (±2.75)	**59.96 (±2.23)**
	100	59.55 (±2.47)	**63.39 (±1.44)**	52.24 (±2.92)	**62.57 (±1.96)**
	250	63.88 (±1.61)	**64.41 (±2.69)**	58.41 (±2.45)	**62.82 (±2.05)**
	500	66.65 (±1.76)	-	62.16 (±2.55)	-
	1000	68.69 (±1.95)	-	63.51 (±1.57)	-
JAFFE	50	52.1 (±2.25)	**74.1 (±1.44)**	36.95 (±3.02)	**81.1 (±2.46)**
	100	62.81 (±1.66)	**83.67 (±1.35)**	48.05 (±2.79)	**82.71 (±2.39)**
	250	73.62 (±2.42)	**87.38 (±1.65)**	63.81 (±2.12)	**84.76 (±2.21)**
	500	79.57 (±1.41)	-	73.33 (±2.55)	-
	1000	83.38 (±1.86)	-	80.24 (±2.26)	-

Fig. 6. *Facial images depicting a person of the JAFFE dataset. From left to right: neutral, anger, disgust, fear, happy, sad and surprise.*

cases of random input weights assignment and low-rank decomposition of the kernel matrix **K**. As can be seen in this Table, the adoption of low-rank decomposition of **K** generally provides enhanced performance, when compared to the random assignment choice for both the ELM and S-ELM algorithms. In addition, ELM and S-ELM networks trained by using the low-rank decomposition choice for input weights determination seem to be more robust, since the corresponding standard deviations over all the experiments for different ELM dimensionalities are smaller.

In our second set of experiments, we have applied the algorithms on the facial image datasets. Grayscale facial images with intensity values in $(0, 1)$ have been employed to this end. Since there is no widely adopted experimental protocol for these datasets too, we also perform the five-fold cross-validation procedure (Devijver and Kittler (1982)), by taking into account the class labels of the data (similar to the experiments conducted on the UCI datasets).

Tables 3 and 4 illustrate the performance of the ELM and S-ELM algorithms for different hidden layer dimensionalities L for the cases of random input weights assignment and low-rank decomposition of the kernel matrix **K** on the face recognition and facial expression recognition datasets, respectively. Similar to the results obtained for the UCI datasets, the adoption of low-rank decomposition of **K** for input weight determination generally provides enhanced performance, when compared to the random assignment choice, for both the ELM and S-ELM algorithms.

7. Conclusions

In this paper, we discussed the connection of the kernel versions of the ELM classifier with infinite Single-hidden Layer Feedforward Neural networks and showed that the original ELM kernel definition can be adopted for the calculation of the ELM kernel matrix for the RBF and sigmoid hidden layer activation functions. In addition, we showed that a low-rank decomposition of the kernel matrix defined on the input training data can be exploited in order to determine an appropriate ELM space for input data mapping, which can be subsequently used for ELM network training. Experimental results denote that the adoption of the low-rank decomposition-based ELM space determination generally leads to enhanced performance, when compared to the standard choice, i.e., random input weights generation.

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

Bai, Z., Huang, G., Wang, W., Wang, H., Westover, M., 2014. Sparse extreme learning machine for classification. IEEE Transactions on Cybernetics accepted.

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.

Deng, W., Zheng, Q., Chen, L., 2004. Regularized extreme learning machine. IEEE Symposium of Computational Intelligence and Data Mining , 389–395.

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.

Gu, S., Guo, Y., 2012. Learning svm classifiers with indefinite kernels. AAAI Conference on Artificial Intelligence .

Helmy, T., Rasheed, Z., 2009. Multi-category bioinformatics dataset classification using extreme learning machine. IEEE Evolutionary Computation , 3234–3240.

Huang, G., 2013. Extreme learning machine. Springer .

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 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.

Kanade, T., Tian, Y., Cohn, J., 2000. Comprehensive database for facial expression analysis. IEEE International Conference on Automatic Face and Gesture Recognition , 46–53.

Lan, Y., Soh, Y., Huang, G., 2008. Extreme learning machine based bacterial protein subcellular localization prediction. IEEE International Joint Conference on Neural Networks , 1859–1863.

Lee, K., Ho, J., Kriegman, D., 2005. Acquiriing linear subspaces for face recognition under variable lighting. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 684–698.

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 sequantial learning algorithm for feedforward networks. IEEE Transactions on Neural Networks 17, 1411–1423.

Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J., 1998. Coding facial expressions with gabor wavelets. IEEE International Conference on Automatic Face and Gesture Recognition , 200–205.

Martinez, A., Kak, A., 2001. Pca versus lda. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 228–233.

Miche, Y., van Heeswijk, M., Bas, P., Simula, O., Lendasse, A., 2011. TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing 74, 2413–2421.

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.

Minhas, R., Baradarani, A., Seifzadeh, S., Jonathan Wu, Q., 2010. Human action recognition using extreme learning machine based on visual vocabularies. Neurocomputing 73, 1906–1917.

Neal, R., 1996. Bayesian learning for neural networks. Lecture Notes in Statistics .

Parviainen, E., Riihimki, J., Miche, Y., Lendasse, A., 2010. Interpreting extreme learning machine as an approximation to an infinite neural network. Knowledge Discovery and Information Retrieval , 1–8.

Pekalska, E., Haasdonk, B., 2009. Kernel discriminant analysis for positive definite and indefinite kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence 31, 2017–1031.

Rong, H., Huang, G., Ong, Y., 2008. Extreme learning machine for multicategories classification applications. IEEE International Joint Conference on Neural Networks , 1709–1713.

Samaria, F., Harter, A., 1994. Parameterisation of a stochastic model for human face identification. IEEE Workshop on Applications of Computer Vision , 138–142.

Smola, A., Ovari, Z., Williamson, R., 2000. Regularization with dot-product kernels. Advances in Neural Information Processing Systems , 308–314.

Suresh, S., Saraswathi, S., Sundararajan, N., 2010. Performance enhancement of Extreme Learning Machine for multi-category sparse data classification problems. Engineering Applications of Artificial Intelligence 23, 1149–1157.

Viangteeravat, T., Shirkhodaie, A., Rababaah, H., 2007. Multiple target vehicles detection and classification based on low-rank decomposition. IEEE International Conference on System of Systems Engineering , 1–8.

Wang, Y., Cao, F., Yuan, Y., 2011. A study on effectiveness of extreme learning machine. Neurocomputing 74, 2483–2490.

Williams, C., 1998. Computation with infinite neural networks. Neural Computation 10, 1203–1216.

Yin, L., Wei, X., Sun, Y., Wang, J., Rosato, M., 2006. A 3d facial expression database for facial behavior research. IEEE International Conference on Automatic Face and Gesture Recognition , 211–216.

Zhang, R., Lan, Y., Huang, G., Zu, Z., 2012. Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Transactions on Neural Networks and Learning Systems 23, 365–371.