A BAYESIAN APPROACH FOR EXTREME LEARNING MACHINE-BASED SUBSPACE LEARNING

Alexandros Iosifidis and Moncef Gabbouj

Department of Signal Processing, Tampere University of Technology, Finland
{alexandros.iosifidis,moncef.gabbouj}@tut.fi

ABSTRACT

In this paper, we describe a supervised subspace learning method that combines Extreme Learning methods and Bayesian learning. We approach the standard Extreme Learning Machine algorithm from a probabilistic point of view. Subsequently and we devise a method for the calculation of the network target vectors for Extreme Learning Machine-based neural network training that is based on a Bayesian model exploiting both the labeling information available for the training data and geometric class information in the feature space determined by the network's hidden layer outputs. We combine the derived subspace learning method with Nearest Neighbor-based classification and compare its performance with that of the standard ELM approach and other standard methods.

Index Terms— Subspace Learning, Network targets determination, Extreme Learning Machine.

1. INTRODUCTION

Feedforward neural networks have been adopted in many real-world problems due to their effectiveness in learning complex classification functions and models. Extreme Learning Machine (ELM) is an algorithm for training Single-hidden Layer Feedforward Neural (SLFN) networks [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. The network output weights are, subsequently, analytically calculated. Similar approaches have been also shown to be efficient in several neural network training methods [2–5], as well as in other learning processes [6]. 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 [7, 8]. Due to its efficiency, ELM networks have 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, mainly targeting supervised and semi-supervised classification [9–12].

Recently, the ELM algorithm has been extended to unsupervised subspace learning [13], where it has been shown that ELM-based unsupervised subspace learning is able to achieve good performance, and compete several state-of-the-art unsupervised subspace learning methods. However, such a subspace learning approach is not able to exploit the additional information of the data labels that is usually available in the training phase. Relevant work in Discriminant Analysis-based subspace learning, e.g. [14, 15], has shown that the exploitation of labeling information is important for the determination of a subspace for data projection that leads to satisfactory classification performance. This paper describes a method that is able to exploit labeling information for ELM-based subspace learning.

In this paper, we first approach the standard ELM approach from a probabilistic point of view. Then, we describe a Bayesian model for the determination of appropriate network target vectors, which is further employed for ELM-based supervised subspace learning. We show that this approach can be exploited by several ELM variants targeting supervised SLFN networks training, e.g. [1, 10, 12, 16], in order to learn a feature space of reduced dimensionality for data projection. We combine it with a simple Nearest Neighbor classifier and compare its performance with that of the standard ELM approach, as well as with the performance of standard subspace learning and classification methods. Experimental results on standard classification problems show that the adoption of the modified network target vectors can enhance the performance of ELM networks and provide performance comparable with that of other subspace learning methods.

The remainder of the paper is organized as follows. Related work in ELM-based SLFN network training is described in Section 2. Our probabilistic model for network target vectors calculation is described in Section 3. Experiments are provided in Section 4. Finally, this work concludes in Section 5.

2. RELATED WORK

Let us denote by $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$ a set of N vectors that we would like to use in order to train a SLFN network. Each vector $\mathbf{x}_i$ belongs to one of the C classes forming our class set and this information is stored to the corresponding

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

class label c_i. For such a classification problem, the adopted SLFN 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 is a parameter of the algorithm and is usually selected to be much greater than the number of classes, i.e., $L \gg C$ [16].

In ELMs, 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 j-th column of $\mathbf{W}_{in}$, the k-th column of $\mathbf{W}_{out}$ and the j-th 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 $\mathbf{o}_i \in \mathbb{R}^C$ of the network corresponding to $\mathbf{x}_i$ is given by:

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

The network hidden layer outputs can be calculated by using almost any nonlinear piecewise continuous activation function $\Phi(\cdot)$, such as the sigmoid, sine and Radial Basis Function (RBF) [8,16,17]. Let us denote by $\boldsymbol{\phi}_i \in \mathbb{R}^L, i = 1, \ldots, N$ the training data representations in the feature space determined by network's hidden layer outputs, noted as ELM space hereafter. (1) can be expressed in a matrix form as $\mathbf{O} = \mathbf{W}_{out}^T\boldsymbol{\Phi}$, where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, $\mathbf{O} \in \mathbb{R}^{C\times N}$ is a matrix containing the network responses for all training data $\mathbf{x}_i$ and the subscript T denotes the transpose operator.

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

$$\mathbf{W}_{out} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T\right)^{-1}\boldsymbol{\Phi}\mathbf{T}^T. \tag{2}$$

The calculation of the network output weights $\mathbf{W}_{out}$ through (2) 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 [16], where the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:}\quad \mathcal{J} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{3}$$

$$\textbf{Subject to:}\quad \mathbf{W}_{out}^T\boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i,\ \ i = 1, ..., N, \tag{4}$$

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 [18], the network output weights $\mathbf{W}_{out}$ can be determined by solving the following equivalent dual optimization problem:

$$\tilde{\mathcal{J}} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 - \sum_{i=1}^{N}\mathbf{a}_i\left(\mathbf{W}_{out}^T\boldsymbol{\phi}_i - \mathbf{t}_i + \boldsymbol{\xi}_i\right). \tag{5}$$

By determining the saddle points of $\tilde{\mathcal{J}}$ with respect to $\mathbf{W}_{out}$, $\boldsymbol{\xi}_i$ and $\mathbf{a}_i$, the network output weights $\mathbf{W}_{out}$ are obtained by:

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

or

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

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *ELM kernel matrix*, having elements equal to $[\mathbf{K}]_{i,j} = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_j$ [19]. In [16] it has been shown that, by exploiting the kernel trick [20], $\mathbf{K}$ can be any positive semidefinite kernel matrix defined over the input data $\mathbf{x}_i$. In [21] it has also been shown that kernel ELM formulations have connections with infinite neural networks. By using (7), the response of the network for a vector $\mathbf{x}_t \in \mathbb{R}^D$ is given by $\mathbf{o}_t = \mathbf{W}_{out}^T\boldsymbol{\phi}_t$. Using (7), one can also define a kernel formulation for the network response calculation, i.e.:

$$\mathbf{o}_t = \mathbf{T}\left(\mathbf{K} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{k}_t, \tag{8}$$

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

Most of the extensions of the ELM algorithm have been proposed for supervised and semi-supervised classification [10–12], trying to design good optimization problems for the calculation of the network output weights. Some others exploit Linear Algebra properties in order to determine a good set of weights and bias values for the hidden layer neurons. For supervised classification, 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 $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. For semi-supervised classification, the target vectors of the labeled training vectors are set as above and 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.

3. TARGET VECTORS DETERMINATION

Here we describe a method for the determination of network target vectors exploiting both training data labeling information and training data geometric information in the ELM space. Due to space limitations, an explanation on why we

expect that the adoption of such target vectors will lead to better ELM-based subspace learning is left for the journal version of the paper.

First, we show that, in the supervised case, the solution obtained for ELM networks employing target values $t_{ik} \in \{-1, 1\}$ is equivalent to the solution obtained for ELM networks employing target values $\tilde{t}_{ik} \in \{0, 1\}$. Let us denote by $\tilde{\mathbf{T}}$ a matrix containing the target vectors $\tilde{\mathbf{t}}_i$ and by $\tilde{\mathbf{W}}_{out}$ the output weights obtained by using $\tilde{\mathbf{T}}$. Let us also denote by $\mathbf{\Phi}^\dagger$ the generalized pseudo-inverse of $\mathbf{\Phi}^T$. The network output weights obtained by using $\mathbf{T}$ are given by:

$$\mathbf{W}_{out} = \mathbf{\Phi}^\dagger \mathbf{T}^T = \mathbf{\Phi}^\dagger (2\tilde{\mathbf{T}} - \mathbf{1})^T = 2\tilde{\mathbf{W}}_{out} - \mathbf{\Phi}^\dagger \mathbf{1}^T, \quad (9)$$

where $\mathbf{1} \in \mathbb{R}^{C \times N}$ is a matrix of ones. The network outputs corresponding to the weights $\tilde{\mathbf{W}}_{out}$ are given by:

$$\tilde{\mathbf{O}} = \tilde{\mathbf{W}}_{out}^T \mathbf{\Phi} = \frac{1}{2}(\mathbf{W}_{out} + \mathbf{\Phi}^\dagger \mathbf{1}^T)^T \mathbf{\Phi} = \frac{1}{2}\mathbf{O} + \frac{1}{2}\mathbf{1}. \quad (10)$$

Thus, in supervised ELMs, employing target values $t_{ik} \in \{-1, 1\}$ is equivalent to using target values $\tilde{t}_{ik} \in \{0, 1\}$. In the latter case, it can be considered that $\tilde{t}_{ik}$ expresses the probability of $\boldsymbol{\phi}_i$ to belong to class k, given the observation c_i, i.e., $t_{ik} \approx p(k|c_i)$. That is, in standard ELMs the determination of the network target vectors exploits only the labeling information available for the training vectors and does not take into account information appearing in $\boldsymbol{\phi}_i$.

Taking into account both c_i and $\boldsymbol{\phi}_i$, the probability of class k can be expressed using the Bayes formula as:

$$p(k|c_i, \boldsymbol{\phi}_i) \propto p(c_i, \boldsymbol{\phi}_i|k)p(k), \quad (11)$$

where $\propto$ denotes the proportionality operator. The class probability $p(k)$ can be estimated by exploiting the labeling information of the training data as $p(k) = \frac{N_k}{N}$, where N_k denotes the number of training vectors belonging to class k. Since for the calculation of the conditional probabilities $p(c_i, \boldsymbol{\phi}_i|k)$ using the training vectors an enormous training set would be required, we can define $p(c_i, \boldsymbol{\phi}_i|k) = p(\boldsymbol{\phi}_i|k)p(c_i|k)$, where independence between c_i and $\boldsymbol{\phi}_i$ is assumed. That is, the probability of class k, given the observations c_i and $\boldsymbol{\phi}_i$, can be expressed by:

$$p(k|c_i, \boldsymbol{\phi}_i) \propto p(\boldsymbol{\phi}_i|k)p(c_i|k)p(k). \quad (12)$$

In order to define the above mentioned probabilities, we would like to define an appropriate model for the classes forming the classification problem, when represented in the ELM space. Let us consider the One-Versus-All classification problem corresponding to the k-th output neuron. By setting the target values $t_{ik} = 1$ for training samples belonging to class k and $t_{ik} = -1$ for training samples that do not belong to class k, ELM approaches assume that class k can be discriminated from all other classes by using a hyperplane passing through the origin in the high-dimensional ELM space $\mathbb{R}^L$. This leads to the assumption that class k is homogeneous in $\mathbb{R}^L$.

Exploiting the class unimodality assumption of ELMs, we define the representation of class k in $\mathbb{R}^L$ by the corresponding class mean vector:

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{j, c_j = k} \boldsymbol{\phi}_j. \quad (13)$$

After the calculation of the class mean vectors $\mathbf{m}_k$, $k = 1, \ldots, C$ of all the classes forming the classification problem, the conditional probabilities $p(\boldsymbol{\phi}_i|k)$ can be calculated by:

$$p(\boldsymbol{\phi}_i|k) = \frac{d(\mathbf{m}_k, \boldsymbol{\phi}_i)}{\sum_{j=1}^{C} d(\mathbf{m}_j, \boldsymbol{\phi}_i)}, \quad (14)$$

where $d(\mathbf{m}_k, \boldsymbol{\phi}_i)$ is a function denoting the similarity of $\boldsymbol{\phi}_i$ and $\mathbf{m}_k$. In order to increase class discrimination, we employ the following similarity measure in our experiments:

$$d(\mathbf{m}_k, \boldsymbol{\phi}_i) = exp\left(-\gamma \|\mathbf{m}_k - \boldsymbol{\phi}_i\|_2^2\right). \quad (15)$$

In (15), $\gamma > 0$ is a parameter that is used in order to scale the Euclidean distance between $\mathbf{m}_k$ and $\boldsymbol{\phi}_i$. In this paper we follow a per-class approach. That is, we determine C parameter values $\gamma_k, k = 1, \ldots, C$, each for one class. We set $\gamma_k = \frac{1}{r\sigma_k^2}$, where σ_k is set equal to the mean Euclidean distance between $\boldsymbol{\phi}_i$, $1 = 1, \ldots, N$ and $\mathbf{m}_k$.

Following a similar approach, the conditional probabilities $p(c_i|k)$ can be obtained by:

$$p(c_i|k) = \frac{d(\mathbf{m}_k, \mathbf{m}_{c_i})}{\sum_{j=1}^{C} d(\mathbf{m}_j, \mathbf{m}_{c_i})}, \quad (16)$$

where $d(\mathbf{m}_k, \mathbf{m}_{c_i})$ is given by:

$$d(\mathbf{m}_k, \mathbf{m}_{c_i}) = exp\left(-\delta \|\mathbf{m}_k - \mathbf{m}_{c_i}\|_2^2\right). \quad (17)$$

While the parameter $\delta > 0$ can be optimized by applying the cross-validation approach, we have experimentally observed that a value of $\delta = \frac{1}{2\sigma_m^2}$, where σ_m is the mean Euclidean distance between the mean class vectors provides satisfactory performance in all tested cases.

In the case where the network hidden layer outputs $\boldsymbol{\phi}_i$, $i = 1, \ldots, N$ are available, the network target vectors can be directly computed by using the previously defined probability values, i.e., $t_{ik} = p(k|c_i, \boldsymbol{\phi}_i)$. In the case where a kernel ELM formulation is adopted, as in (7), the distance values in (14) and (15) can be calculated by employing the pairwise training vectors similarity values stored in the ELM kernel matric $\mathbf{K}$. The target vectors $\mathbf{t}_i$ can be normalized in order to have unit l_1 norm. However, we have experimentally found that this is not necessary. It should be noted here that the standard ELM approach using target values $t_{ik} \in \{-1, 1\}$ is a special case of the previously described approach for $\gamma \ll 1$, $\delta \gg 1$ and $p(k) = 1$, $k = 1, \ldots, C$.

Table 2. Mean classification rates (%) and standard deviations on standard data sets.

	ELM		RELM		KELM			
Data set	Standard	**Modified**	Standard	**Modified**	Standard	**Modified**	SVM	KSR
Column	78.19 (±0.91)	**78.71 (±0.49)**	79.45 (±0.59)	**80.74 (±0.22)**	81.03 (±0.2)	**81.13 (±0.51)**	**81.35 (±0.2)**	77.68 (±0.82)
Glass	63.69 (±1.3)	**65.68 (±0.9)**	64.92 (±0.65)	**66.76 (±0.77)**	68.45 (±0.73)	**69.43 (±0.7)**	**71.41 (±0.31)**	64.25 (±0.78)
Ionosphere	93.02 (±0.76)	**94.19 (±0.54)**	93.02 (±0.76)	**94.27 (±0.47)**	95.07 (±0.27)	**95.36 (±0.27)**	**95.95 (±0.18)**	95.24 (±0.63)
Iris	95.53 (±0.63)	**97.13 (±0.32)**	95.53 (±0.63)	**97.73 (±0.72)**	97.4 (±0.21)	**97.47 (±0.42)**	97.33 (±0.01)	94.8 (±0.42)
Libras	64.39 (±0.42)	**82.28 (±0.93)**	64.87 (±0.47)	**84.59 (±0.41)**	85.26 (±0.3)	**85.71 (±0.94)**	85.55 (±0.34)	**86.21 (±0.2)**
Madelon	57.76 (±0.6)	**58.25 (±0.56)**	60.01 (±0.33)	**60.73 (±0.43)**	63.73 (±0.13)	**64.51 (±0.21)**	64.15 (±0.13)	63.59 (±0.01)
Segmentation	88.78 (±0.01)	**94.94 (±0.13)**	89.97 (±0.11)	**95.08 (±0.15)**	95.68 (±0.01)	**96.57 (±0.01)**	96.9 (±0.01)	**97.45 (±0.01)**
Synth.Control	88.68 (±0.2)	**93.27 (±0.56)**	91.87 (±0.32)	**93.95 (±0.39)**	95.25 (±0.26)	**96.48 (±0.01)**	**96.93 (±0.21)**	95.33 (±0.01)
Tae	47.27 (±0.57)	**65.77 (±0.94)**	50.74 (±0.47)	**68.75 (±0.97)**	54.26 (±0.01)	**67.45 (±0.01)**	57.08 (±0.01)	60.2 (±0.01)
TicTacToe	83.09 (±0.01)	**85.5 (±1.03)**	86.34 (±0.29)	**92.68 (±0.53)**	97.29 (±0.01)	**97.39 (±0.01)**	91.23 (±0.01)	83.72 (±0.01)

Table 1. Data sets used in our experiments.

Data set	Samples	D	C
Column	310	6	3
Glass	214	9	6
Ionosphere	351	34	2
Iris	150	4	3
Libras	360	90	15
Madelon	2600	500	2
Segmentation	2310	19	7
Synth.Control	600	60	6
Tae	151	5	3
TicTacToe	958	9	2

4. EXPERIMENTS

We have applied our method on ten standard classification problems coming from the machine learning repository of the University of California Irvine (UCI) [22]. Information concerning the data sets used is illustrated in Table 1. Due to space limitations, results on more datasets, along with comparisons of the training times in larger data sets between the standard ELM and proposed approaches and a statistical significance analysis of the experimental results are left for the journal version of the paper.

For each data set we apply 10 experiments and measure the performance of the method by calculating the mean classification rate over all experiments and the corresponding standard deviation value. In each experiment, the five-fold cross-validation procedure is applied, where we take into account the class labels of the data. This means that on each experiment we randomly split each class in five sets and apply five training-evaluation rounds by using the obtained set indices.

We compare the performance of ELM [1], RELM [16] and KELM [16] networks described in Section 2, when using the standard target vectors and the ones determined by applying our method. For the KELM method exploiting a kernel formulation we have employed the RBF kernel function, where we set the parameter σ equal to the mean Euclidean distance between the training vectors $\mathbf{x}_i$, which corresponds to the natural scaling factor for each data set. For the ELM methods using random hidden weights, we used the RBF activation function $\Phi(\mathbf{v}_j, b_j, \mathbf{x}_i) = exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{v}_j\|_2^2}{2b_j^2}\Big)$, where the value b_j is set equal to the mean Euclidean distance between the training data $\mathbf{x}_i$ and the network input weights $\mathbf{v}_j$. The optimal value of the parameters r and c have been determined by using a grid search strategy with values $r = 10^{-3,\ldots,3}$ and $c = 10^{0,\ldots,6}$, respectively. For ELM methods employing random hidden weights we used $L = 1000$ hidden layer neurons, which is a value that has been shown to provide good results in many classification problems [16].

Table 2 illustrates the performance of each algorithm, when using the standard and the modified approaches for the determination of the network's target vectors. The adoption of our method enhances the performance of ELM, RELM and KELM networks in all cases. In Table 2 we also provide the performance obtained by applying Support Vector Machine (SVM)-based and Kernel Spectral Regression (KSR)-based classification. We omit providing the performance of Unsupervised ELM (UELM), since its performance was far inferior when compared to the remaining methods. This is reasonable, since UELM does not take into account class information of the training data. As can be seen in Table 2 there is not a widely optimal classification scheme. KSR-based classification achieves the best performance in two out of ten cases, SVM achieves the best performance in four, while the KELM-based method using the modified target vectors provides the best performance in five out of ten cases.

5. CONCLUSIONS

In this paper, a supervised subspace learning method has been described and evaluated. The method builds a Bayesian model that exploits the assumptions of the Extreme Learning Machine algorithm for the calculation of network target vectors. These vectors exploit information relating to both the training class labels, as well as geometric class information

in the ELM space. Our approach can be exploited by several ELM variants proposed for supervised neural network training in order to learn a feature space for nonlinear data projection and classification.

Acknowledgment

This work was supported by the Finnish Funding Agency for Technology and Innovation (TEKES) as part of the Visual label project with A-Lehdet and the Data to Intelligence program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the eld of ICT and digital business).

REFERENCES

[1] G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," *IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 985–990, 2004.

[2] W.F. Schmidt, M.A. Kraaijveld, and R.P.W. Duin, "Feedforward neural networks with random weights," *International Conference on Pattern Recognition*, 1992.

[3] Y.H. Pao, G.H. Park, and D.J. Sobajic, "Learning and generalization characteristics of random vector functional-link net," *Neurocomputing*, vol. 6, pp. 163–180, 1994.

[4] C.L.P. Chen, "A rapid supervised learning neural network for function interpolation and approximation," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1220–1230, 1996.

[5] B. Widrow, A. Greenblatt, Y. Kim, and D. Park, "The no-prop algorithm: A new learning algorithm for multilayer neural networks," *Neural Networks*, vol. 37, pp. 182–188, 2013.

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

[7] R. Zhang, Y. Lan, G.B. Huang, and Z.B. Zu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 365–371, 2012.

[8] G.B. Huang, L. Chen, and C.K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[9] G.B. Huang, D. Wang, and Y. Lan, "Extreme learning machine: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[10] J. Luo, C.M. Vong, and P.K. Wong, "Sparse bayesian extreme learning machine for multi-classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 836–843, 2014.

[11] A. Iosifidis, A. Tefas, and I. Pitas, "Regularized extreme learning machine for multi-view semi-supervised action recognition," *Neurocomputing*, vol. 145, pp. 250–262, 2014.

[12] N. Wang, M.J. Er, and M. Han, "Parsimonious extreme learning machine using recursive orthogonal least squares," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1828–1841, 2014.

[13] G. Huang, S. Song, J.N.D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[14] A.M. Martinez and A.C. Kak, "PCA versus LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[15] A. Iosifidis, A. Tefas, and I. Pitas, "On the optimal class representation in linear discriminant analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1491–1497, 2013.

[16] G.B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[17] G.B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2008.

[18] R. Fletcher, *Practical Methods of Optimization: Volume 2 Constrained Optimization*, Wiley, 1981.

[19] B. Frenay and M. Verleysen, "Using svms with randomised feature spaces: An extreme learning approach," *European Symposium of Artificial Neural Networks*, 2010.

[20] B. Scholkpf and A.J. Smola, "Learning with kernels," 2001, MIT Press.

[21] A. Iosifidis, A. Tefas, and I. Pitas, "On the kernel extreme learning machine classifier," *Pattern Recognition Letters*, vol. 54, pp. 11–17, 2015.

[22] K. Bache and M. Lichman, "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science," 2013.