Iosifidis, A., Tefas, A., & Pitas, I. (2016). Large-scale nonlinear facial image classification based on approximate kernel Extreme Learning Machine. In 2015 IEEE International Conference on Image Processing (ICIP 2015): Proceedings of a meeting held 27-30 September 2015, Quebec City, Quebec, Canada. (pp. 2449-2453). Institute of Electrical and Electronics Engineers (IEEE). DOI: 10.1109/ICIP.2015.7351242

LARGE-SCALE NONLINEAR FACIAL IMAGE CLASSIFICATION BASED ON APPROXIMATE KERNEL EXTREME LEARNING MACHIINE

Alexandros Iosifidis, Anastasios Tefas and Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
Email: {tefas,pitas}@aiia.csd.auth.gr

ABSTRACT

In this paper, we propose a scheme that can be used in large-scale nonlinear facial image classification problems. An approximate solution of the kernel Extreme Learning Machine classifier is formulated and evaluated. Experiments on two publicly available facial image datasets using two popular facial image representations illustrate the effectiveness and efficiency of the proposed approach. The proposed Approximate Kernel Extreme Learning Machine classifier is able to scale well in both time and memory, while achieving good generalization performance. Specifically, it is shown that it outperforms the standard ELM approach for the same time and memory requirements. Compared to the original kernel ELM approach, it achieves similar (or better) performance, while scaling well in both time and memory with respect to the training set cardinality.

Index Terms— Nonlinear Facial Image Classification, Extreme Learning Machine, Approximate Methods

1. INTRODUCTION

Extreme Learning Machine (ELM) [1] is an algorithm for Single-hidden Layer Feedforward Neural (SLFN) networks training that has attracted considerable research attention in the last decade due to its effectiveness and efficiency in many medium-scale pattern classification problems [2, 3]. The idea of exploiting randomness in neural network training processes has been proposed in early 1990's [4, 5, 6, 7] and has rejuvenated in middle 2000's [1, 8] under the term Extreme Learning Machine (ELM). Algorithms following this approach set the assumption that the learning processes adopted for the determination of 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, this way defining a random (nonlinear) mapping of the input space to a new (usually high-dimensional) feature space.

By using a large number of (independent) hidden layer weights, it is expected that the problem to be solved is transformed to a linear problem in the new feature space and, thus, linear techniques like mean square estimation can be employed for the determination of the network's output weights. Specifically, in ELM approaches, the network input weights are randomly assigned, while the network output weights are analytically calculated so as to minimize the network training error. Randomized ELMs not only tend to reach the smallest training error, but also the smallest output weight norm as well. For feedforward networks reaching a small training error, smaller output weight norm results in better generalization performance [9]. It has been proven that, by using an adequate number of hidden layer neurons, randomized ELMs have the properties of global approximators [10, 8].

Recently, kernel versions of the ELM algorithm have been proposed [11, 12, 13], which have been shown to outperform the standard ELM approach that uses random input parameters. A possible explanation of this fact is that kernel ELM networks have connections to infinite single-hidden layer feedforward networks [14]. However, the superior performance of kernel ELMs comes with a higher computation cost and memory requirements. Specifically, kernel ELMs require the calculation of the corresponding kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, having a quadratic computational complexity with respect to the number of training data N, while the calculation of the network parameters requires the inversion of $\mathbf{K}$, having a cubic computational complexity with respect to N, i.e. $O(N^3)$. This fact renders the application of kernel ELMs in large scale classification problems prohibitive.

In order to make the application of ELM-based classification in large scale classification problems possible, an SMO-based optimization algorithm has been proposed in [12] for the case where the hinge loss of the prediction error is exploited. This method has the drawbacks that it still requires the calculation of the entire kernel matrix $\mathbf{K}$ and that the optimization process is still very slow for large scale datasets. For squared loss criteria, the randomized ELM approach has been proposed in [11]. It has the disadvantage that, by employing randomly sampled hidden layer parameters, the obtained performance is inferior, when compared to the kernel approach [14].

Approximation approaches have been found to be both efficient and effective. A line of work in approximate methods determines a low-rank approximation of a Gram matrix of the form $\mathbf{K} \simeq \mathbf{Q} = \mathbf{C}\tilde{\mathbf{Q}}\mathbf{C}^T$, where $\mathbf{C} \in \mathbb{R}^{N \times n}$ and $\tilde{\mathbf{Q}} \in \mathbb{R}^{n \times n}$. $\mathbf{C}$ is formed by n (uniformly or data-dependent nonuniformly sampled) columns of $\mathbf{K}$ and $\tilde{\mathbf{Q}}$ is a matrix constructed by the intersection between those n columns of $\mathbf{K}$ and the corresponding n rows of $\mathbf{K}$ [15]. By using such matrix approximation approaches, approximate Linear Algebra methods, like matrix multiplication and Singular Value Decomposition, and their application in kernel machines have been proposed that have provable guarantees on the quality of obtained approximate solution [16, 17, 18, 19, 15]. In addition, it has been recently shown that the adoption of such an approximate approach can be exploited in kernel-based clustering [20, 21, 22] with state-of-the-art performance. While the above-described approximate approach has the advantage that the entire kernel matrix needs not be computed, for KELM-based classification the the inversion of the corresponding approximate kernel matrix $\mathbf{Q} \in \mathbb{R}^{N \times N}$ still has a computational complexity equal to $O(N^3)$. Another approximate approach exploits a so-called "randomized kernel", which is constructed by randomly sampling a small set n of the N training data [18, 23, 24]. This approach has the disadvantage that the corresponding methods exploit information appearing only in the subset of the training data employed for the calculation of the randomized kernel.

In this paper, we propose a novel kernel ELM approximation (noted as AKELM hereafter). We show that the proposed approach is able to scale well in memory and operates very fast, when compared to the kernel ELM approach, while achieving comparable, or even better, performance with that of kernel versions of ELM. We evaluate the proposed approach in medium and large scale classification problems, where we compare its performance with that of randomized ELMs [1, 11], kernel ELM applied on the entire training set [11] and kernel ELM that exploits a randomized kernel obtained by using $\tilde{N} < N$ samples, similar to [18, 23, 24].

2. EXTREME LEARNING MACHINES

In this section, we briefly describe the ELM, regularized ELM and kernel ELM algorithms proposed in [1] and [11]. Let us denote by $\mathbf{X}$ a set of N vectors $\mathbf{x}_i \in \mathbb{R}^D$ and by c_i the corresponding class labels ($c_i \in \{1, \ldots, C\}$). An ELM network is essentially a combination of C one-versus-rest classifiers. It consists of D input, L hidden and C output 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 $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. In randomized 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.

Let us denote by $\mathbf{q}_j$, $\mathbf{w}_k$, w_{kj} the j-th column of $\mathbf{W}_{in}$, the k-th row 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 and using a linear activation function for the network output layer, the response $\mathbf{o}_i = [o_{i1}, \ldots, o_{iC}]^T$ of the network corresponding to $\mathbf{x}_i$ is calculated by:

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

It has been shown that almost any nonlinear piecewise continuous activation functions $\Phi(\cdot)$ can be used for the calculation of the network hidden layer outputs, e.g. the sigmoid, sine, Gaussian, hard-limiting, Radial Basis Function (RBF), RBF-χ^2, Fourier series, etc [8, 25, 11, 26, 27, 28]. By storing the network hidden layer outputs $\boldsymbol{\phi}_i \in \mathbb{R}^L$ corresponding to all the training vectors $\mathbf{x}_i$, $i = 1, \ldots, N$ in a matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N]$, equation (1) can be expressed in a matrix form as:

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

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

By assuming that $\mathbf{o}_i = \mathbf{t}_i$, $i = 1, \ldots, N$, or in 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 $\mathbf{W}_{out}$ can be analytically calculated by:

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

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

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l, \tag{4}$$

where $\boldsymbol{\phi}_l$ is the network hidden layer output for $\mathbf{x}_i$.

The calculation of the network output weights $\mathbf{W}_{out}$ through (3) is sometimes inaccurate, since the matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ may be 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 [11], where the network output weights are calculated by solving the following optimization problem:

$$\textbf{Minimize:} \quad \mathcal{J}_{RELM} = \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{\lambda}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \tag{5}$$

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \boldsymbol{\phi}_i = \mathbf{t}_i - \boldsymbol{\xi}_i, \ i = 1, ..., N, \tag{6}$$

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 solving the equivalent to (5) dual optimization problem under the constraints in (6) [29], the network output weights $\mathbf{W}_{out}$ are obtained by:

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

or

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

where $\mathbf{K} \in \mathbb{R}^{N \times N}$ is the *ELM kernel matrix*, having elements equal to $[\mathbf{K}]_{i,j} = \boldsymbol{\phi}_i^T \boldsymbol{\phi}_j$ [30]. By using (8), the network response for a given vector $\mathbf{x}_l \in \mathbb{R}^D$ is given by:

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l = \mathbf{A}\mathbf{k}_l, \tag{9}$$

where $\mathbf{A} = \mathbf{TQ}$, $\mathbf{Q} = \left(\mathbf{K} + \frac{1}{\lambda}\mathbf{I}\right)^{-1}$ and $\mathbf{k}_l \in \mathbb{R}^N$ is a vector having its elements equal to $\mathbf{k}_{l,i} = \boldsymbol{\phi}_i^T \boldsymbol{\phi}_l$, $i = 1, \dots, N$.

3. PROPOSED KERNEL ELM APPROXIMATION

In order to obtain an approximate kernel ELM solution, we assume that the network output weights $\mathbf{W}_{out}$ lie on the span of a subset of the training data (represented in the kernel space $\mathcal{F}$), i.e. $\mathbf{W}_{out} = \tilde{\boldsymbol{\Phi}}\mathbf{A}^T$, where $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{|\mathcal{F}| \times n}$, where $\mathbf{A} \in \mathbb{R}^{C \times n}$ is a matrix containing the reconstruction weights of $\mathbf{W}_{out}$ with respect to $\tilde{\boldsymbol{\Phi}}$. The columns of $\tilde{\boldsymbol{\Phi}}$ are randomly selected from the columns of $\boldsymbol{\Phi}$, i.e.:

$$\tilde{\boldsymbol{\Phi}} = \boldsymbol{\Phi}\mathbf{E}\mathbf{J}, \tag{10}$$

where $\mathbf{E}$ is a random (column) permutation matrix and $\mathbf{J} \in \mathbb{R}^{N \times n}$ is a matrix with elements $J_{ii} = 1$ and $J_{ij} = 0,\ i \neq j$. AKELM solves (2), by setting $\mathbf{O} = \mathbf{T}$. This can be expressed as follows:

$$\mathbf{T} = \mathbf{W}_{out}^T \boldsymbol{\Phi} = \mathbf{A}\tilde{\boldsymbol{\Phi}}^T \boldsymbol{\Phi} = \mathbf{A}\tilde{\mathbf{K}}, \tag{11}$$

where $\tilde{\mathbf{K}} \in \mathbb{R}^{n \times N}$ is a submatrix of the original kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$. The optimal reconstruction weight matrix $\mathbf{A}$ is given by:

$$\mathbf{A} = \mathbf{T}\tilde{\mathbf{K}}^T \left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T\right)^{-1} \tag{12}$$

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

$$\mathbf{o}_l = \mathbf{W}_{out}^T \boldsymbol{\phi}_l = \mathbf{A}\tilde{\boldsymbol{\Phi}}^T \boldsymbol{\phi}_l = \mathbf{A}\tilde{\mathbf{k}}_l. \tag{13}$$

It should be noted here that the calculation of $\mathbf{A}$ through (12) is always possible, since $n \leq N$.

Comparing (13) and (9), it can be seen that the proposed AKELM algorithm has a much lower computational complexity, when compared to the KELM. Specifically, the computational complexity of the KELM algorithm is equal to $O(N^3 + (D + C + 1)N^2)$, while the computational complexity of the proposed AKELM algorithm is equal to $O((2n^2 + nD + C)N)$. By setting $n = pN$ and $D = mN$, the time complexity of the KELM algorithm is equal to $O((m + 1)N^3 + (C + 1)N^2)$ and the time complexity of the proposed AKELM algorithm is equal to $O((2p^2 + p)N^3 + CN)$. Taking into account that, for large scale classification problems, $m \ll 1$ and that satisfactory performance can be achieved by using a value $p \ll 1$, as shown in the experimental evaluation provided in Section 4, the computational cost of the proposed AKELM algorithm in the training phase is significantly lower than the one of the KELM algorithm. For example, in the Youtube Faces database [31], the proposed AKELM algorithms achieves a good performance by using a value of $p = 1.6 \cdot 10^{-3}$. In that database, where $m = 6 \cdot 10^{-3}$, the acceleration achieved by applying the proposed AKELM algorithm versus KELM is in the order of 10^3. In the test phase, the time complexity of KELM algorithm is equal to $O(CN^2 + D^2N) = O(CN^2 + m^2N^3)$, while the time complexity of the proposed AKELM algorithm is equal to $O(Cn^2 + D^2n) = O(Cp^2N^2 + m^2pN^2)$. Thus, in the test phase the computational cost of the proposed AKELM algorithm is significantly lower from that of KELM too. Regarding memory requirements, KELM employs a matrix of $N \times N$ dimensions, while the proposed AKELM a matrix of $pN \times N$ dimensions.

Table 1. Performance on the BU dataset.

L,n,$\tilde{N}$	ELM	KELM	RELM	**AKELM**
25	47.14%	28.43%	47.14%	**47.71**%
50	51.14%	38%	52.29%	**56.43**%
100	60.57%	46.43%	60.86%	**62.43**%
250	67.43%	51.57%	67.86%	**69.29**%
500	67.86%	57.57%	**69.19**%	**69.14**%
14000	-	68.29%	-	-

4. EXPERIMENTS

In this Section, we present experiments conducted in order to evaluate the performance of the proposed AKELM algorithm. We have employed two publicly available facial image datasets to this end, i.e. BU [32] and YouTube Faces [31]. In all the experiments we compare the performance and efficiency of the proposed AKELM algorithms with that of ELM [1], RELM [11] and Kernel ELM (KELM) [11] algorithms. All the experiments have been conducted on a 4-core, $i7 - 4790$, 3.6GHz PC with 32GB RAM using single floating point precision and a MATLAB implementation. For the ELM methods exploiting a kernel formulation we have employed the RBF kernel function, where the value of σ is set equal to the mean Euclidean distance between the training vectors $\mathbf{x}_i$ that corresponds to the natural scaling value for each dataset. In the case of randomized ELMs, we have employed the RBF activation function where the value b is set equal to the mean Euclidean distance between the training data $\mathbf{x}_i$ and the network input weights $\mathbf{q}_j$. The optimal value of the regularization parameter λ used in RELM and KELM algorithms has been determined by applying line search using values $\lambda = 10^r,\ r = -6, \dots, 6$.

On the BU dataset, we have resized the facial images to 40×30 pixel images and vectorized these images in order to create vectors $\mathbf{x}_i \in \mathbb{R}^{1200}$. Since there is not a widely

Table 3. Training and test times and memory during training in BU dataset.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n,$\tilde{N}$	ELM/RELM	KELM	AKELM	ELM/RELM	KELM	AKELM	ELM/RELM	KELM	AKELM
25	0.179	0.001	0.18	0.002	0.001	0.002	2.67	0.006	2.67
50	0.197	0.002	0.198	0.002	0.001	0.002	5.34	0.024	5.34
100	0.237	0.004	0.239	0.002	0.001	0.002	10.68	0.084	10.68
250	0.414	0.018	0.416	0.004	0.003	0.004	26.7	0.484	26.7
500	0.704	0.08	0.706	0.009	0.005	0.009	53.41	1.938	53.41
14000	-	**152.17**	-	-	**1.265**	-	-	**1495.4**	-

Table 4. Training and test times and memory during training in Youtube Faces dataset.

	Training time (sec)			Test time (sec)			Memory (MB)		
L,n/$\tilde{N}$	ELM/RELM	KELM	AKELM	ELM/RELM	KELM	AKELM	ELM/RELM	KELM	AKELM
25/-	3.26	-	3.27	0.78	-	0.78	28.25	-	28.25
50/500	3.57	0.06	3.52	0.84	2.02	0.84	56.51	3.81	56.51
100/1000	4.59	0.22	4.64	0.96	3.21	0.96	113.01	15.26	113.01
250/2000	6.11	1.40	5.91	1.35	5.55	1.35	282.53	95.37	282.53
500/5000	10.01	23.68	9.64	1.95	**13.06**	1.95	565.94	381.47	565.94
1000/10000	21.82	**182.81**	21.51	3.22	**25.85**	3.22	1131.9	1525.4	1131.9
2000/20000	42.96	**1411.8**	41.41	5.63	**50.47**	5.63	2263.9	1525.4	2263.9
5000/296257	322.52	$\mathbf{4 \cdot 10^5}$	322.06	13.07	**820.9**	13.07	4537.6	$\mathbf{3 \cdot 10^5}$	4537.6

Table 2. Performance (CR) on the Youtube Faces dataset.

L,n/$\tilde{N}$	ELM	KELM	RELM	**AKELM**
25/−	7.57%	-	17.9%	**23.93**%
50/−	31.3%	-	31.31%	**36.28**%
100/500	45.18%	67.15%	45.18%	**51.16**%
250/1000	67.32%	76.61%	67.37%	**73.84**%
500/2000	54.72%	78.37%	54.74%	**84.45**%
1000/5000	73.81%	83.14%	73.84%	**85.08**%
2000/10000	77.03%	85.12%	77.03%	**89.93**%
5000/20000	86.61%	90.18%	86.64%	**94.35**%

adopted experimental protocol for this dataset, we perform the five-fold cross-validation procedure [33]. On each cross-validation step, we enrich the training set by following the approach proposed in [34]. The cardinality of the training set used on each cross-validation step is equal to $N = 14000$. On the YouTube Faces dataset, we have employed the LBP-based facial image representation suggested in [31] leading to a facial image representation $\mathbf{x}_i$ of $D = 1770$ dimensions. In these experiments we have employed the facial images depicting persons in at least 500 images, resulting to a dataset of 370319 images and 340 classes. Since there is no widely adopted experimental protocol on the YouTube Faces dataset for multi-class classification, we retain 80% of the facial images for training and the remaining 20% for testing, leading to a training set cardinality equal to $N = 296257$.

In order to test the performance of the original KELM algorithm in these datasets, we adopt training data sampling for the creation of a training set of smaller cardinality $\tilde{N}$ [23, 24]. The performance, the computational cost and memory requirements of each algorithm for different values of $L, n, \tilde{N}$ are illustrated in Tables 1 - 4. As can be seen, the proposed AKELM is both effective and efficient, outperforming the remaining methods in most cases, while requiring a much lower computational cost.

5. CONCLUSIONS

In this paper, we proposed an approximate solution of the kernel Extreme Learning Machine classifier that is able to scale well in both training/test computational cost and memory. Experimental evaluation on facial image classification problems illustrates that the proposed approximate approach is able to operate extremely fast in both the training and test phases and to provide satisfactory performance, outperforming relating classification schemes in most cases.

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).

6. REFERENCES

[1] G.B. Huang, Q.Y. Zhu, and C.K. Siew, “Extreme Learning Machine: a new learning scheme of feedforward neural networks,” *Proc IEEE Int Jt Conf Neural Netw*, vol. 2, pp. 985–990, 2004.

[2] G.B. Huang, “Extreme Learning Machine,” *Springer*, 2013.

[3] G.B. Huang, “An insight into Extreme Learning Machines: Random neurons, random features and kernels,” *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[4] D.S. Broomhead and D. Lowe, “Multivariable functional interpolation and adaptive networks,” *Complex Systems*, vol. 2, pp. 321–355, 1988.

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

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

[7] 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.

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

[9] P.L. Bartlett, “The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network,” *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[10] R. Zhang, Y. Lan, G.B. Huang, and Z.B. Zu, “Universal approximation of Extreme Learning Machine with adaptive growth of hidden nodes,” *IEEE Trans Neural Netw. Learn. Syst.*, vol. 23, no. 2, pp. 365–371, 2012.

[11] G.B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme Learning Machine for regression and multiclass classification,” *IEEE Trans. Syst., Man, Cybern. B, Cybern*, vol. 42, no. 2, pp. 513–529, 2012.

[12] Z. Bai, G.B. Huang, W. Wang, H. Wang, and M.B. Westover, “Sparse Extreme Learning Machine for classification,” *IEEE Trans Cybern.*, vol. 44, no. 10, pp. 1858–1870, 2014.

[13] A. Iosifidis, A. Tefas, and I. Pitas, “Graph embedded extreme learning machine,” *IEEE Transactions on Cybernetics, D.O.I. 10.1109/TCYB.2015.2401973*, 2015.

[14] A. Iosifidis, A. Tefas, and I. Pitas, “On the kernel Extreme Learning Machine classifier,” *Pattern Recogn. Lett.*, vol. 54, pp. 11–17, 2015.

[15] P. Drineas and M.W. Mahoney, “On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-based Learning,” *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2275, 2005.

[16] J. Smola and B. Scholkopf, “Sparse greedy matrix approximation for machine learning,” *Int. Conf. Mach. Learn.*, 2000.

[17] C.K.I. Williams and M. Seeger, “The effect of the input density distribution on kernel-based classifiers,” *Int. Conf. Mach. Learn.*, 2000.

[18] C.K.I. Williams and M. Seeger, “Using the Nystrom method to speed up kernel machines,” *Adv. Neural Inf. Process. Syst.*, 2001.

[19] P. Drineas, R. Kannan, and M.W. Mahoney, “Fast Monte Carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix,” *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, 2006.

[20] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, “Spectral grouping using the Nystrom method,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.

[21] R. Chitta, R. Jin, T.C. Havens, and A.K. Jain, “Approximate kernel k-means: solution to large scale kernel clustering,” *Int. Conf. Knowl. Disc. and Data Mining*, 2011.

[22] P. Drineas, R. Kannan, and M.W. Mahoney, “Scalable kernel clustering: Approximate kernel k-means,” *arXiv:1402.3849v1*, pp. 1–15, 2014.

[23] D. Achilioptas, G. McSherry, and B. Scholkopf, “Sampling techniques for kernel methods,” *Adv. Neural Inf. Process. Syst.*, 2002.

[24] M.A. Belabbas and P.J. Wolfe, “Spectral methods in machine learning and new strategies for very large datasets,” *Proc. Nat. Acad. of Sciences*, vol. 106, no. 2, pp. 369–374, 2009.

[25] G.B. Huang and L. Chen, “Convex incremental Extreme Learning Machine,” *Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2008.

[26] A. Iosifidis, A. Tefas, and I. Pitas, “Minimum class variance Extreme Learning Machine for human action recognition,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 11, pp. 1968–1979, 2013.

[27] A. Iosifidis, A. Tefas, and I. Pitas, “Minimum variance Extreme Learning Machine for human action recognition,” *IEEE Int. Conf. Acoust., Speech and Sign. Proc.*, 2014.

[28] 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.

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

[30] B. Frenay and M. Verleysen, “Using svms with randomised feature spaces: An extreme learning approach,” *Europ. Symp. Art. Neural Netw.*, 2010.

[31] L. Wolf, T. Hassner, and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” *Comp. Vision and Patt. Recogn.*, 2011.

[32] L. Yin, X. Wei, Y. Sun, J. Wang, and M. Rosato, “A 3d facial expression database for facial behavior research,” *IEEE Int. Conf. on Autom. Face and Gest. Recogn.*, 2006.

[33] P. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, 1982.

[34] A. Maronidis, D. Bolis, A. Tefas, and I. Pitas, “Improving subspace learning for facial expression recognition using person dependent and geometrically enriched training sets,” *Neural Networks*, vol. 24, no. 8, pp. 814–823, 2011.