SUPERVISED SUBSPACE LEARNING BASED ON DEEP RANDOMIZED NETWORKS

Alexandros Iosifidis and Moncef Gabbouj

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

ABSTRACT

In this paper, we propose a supervised subspace learning method that exploits the rich representation power of deep feedforward networks. In order to derive a fast, yet efficient, learning scheme we employ deep randomized neural networks that have been recently shown to provide good compromise between training speed and performance. For optimally determining the learnt subspace, we formulate a regression problem where we employ target vectors designed to encode both the labeling information available for the training data and geometric properties of the training data, when represented in the feature space determined by the network's last hidden layer outputs. We experimentally show that the proposed approach is able to outperform deep randomized neural networks trained by using the standard network target vectors.

Index Terms— Supervised Subspace Learning, Deep Neural Networks, Network targets calculation

1. INTRODUCTION

Feedforward neural networks have been adopted in many pattern recognition problems due to their ability to approximate any complex decision function and because they achieve excellent performance in real-world applications. While there are theoretical proofs for the global approximation ability of single-hidden layer networks [1, 2], properly trained deep neural networks can learn much more complex relationships between the training data, leading to feature spaces of higher representation powers. The adoption of such feature spaces has the potential of achieving improved classification performance.

In order to overcome the drawbacks of slow training and early convergence of feedforward networks trained using gradient-based weight adaptation strategies, like the Backpropagation algorithm [3], randomized approaches have been proposed, mainly for single-hidden layer networks [4, 5, 6, 7, 8]. The main idea in such techniques is that the learning process used for the determination of the network's hidden layer and the output weights need not be connected. By employing input weights which are randomly drawn from a multi-dimensional distribution (e.g. Gaussian, or uniform), they apply a randomized data projection to a feature space of (usually) high dimensions (commonly followed by a non-linearity). By using an adequately high number of hidden layer neurons, it has been proven that the problem to be solved can be transformed to an easier one, where linear techniques, like Least Squares-based regression, can be applied. The fact that the network's hidden and output weights are determined independently has a number of advantages that can be exploited, for example, for facilitating the implementation of parallel/distributed systems. In addition, it has been shown to perform well in many classification problems, such as human action recognition [9, 10, 11].

Recently, this approach has been extended for the training of deep neural networks, i.e. networks formed by multiple hidden layers, [12, 13]. These training algorithms, either employ random weights for all the hidden layers of the network, or stack unsupervised representations learned by using random hidden layer neurons for training Auto-Encoders (AE). Finally, linear (Least Squares-based) regression is applied in order to learn the network's output weights based on target vectors, each of which denotes the class of a training sample. This process leads to the determination of more complex and powerful data representations and usually enhances performance, when compared to single-hidden layer networks [12, 13].

In this paper, we are interested in supervised subspace learning approaches that are able to exploit the enriched representation power of deep neural networks. In order to derive a fast, yet efficient, learning scheme we employ the Hierarchical Extreme Learning Machine (H-ELM) algorithm recently proposed in [13] for randomized deep neural networks training that is based on a hierarchic process learning data representations of increased representation power. We show that for effective supervised subspace learning using regression models in deep networks, the network's output weights need to be determined by using modified target vectors encoding both the labeling information of the training data and geometric properties of the feature space determined by the network's last hidden layer outputs. For the determination of such target vectors, we reformulate the method proposed in our recent work [14], which is based on a Bayesian class description. We experimentally show that deep networks trained by using the proposed method are able to outperform deep

networks trained by applying the original approach exploiting the standard network targets [13].

2. RELATED WORK

In this Section, we briefly describe the training algorithm proposed in [13] for deep neural networks' training that we will employ in our supervised subspace learning scheme.

Let us denote by $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$ a set of vectors and the corresponding class labels $c_i \in \{1, \ldots, C\}$. We will use these vectors in order to train a deep neural network formed by N_L hidden layers. The number of neurons in the network's input and output layers are equal to D (equal to the dimensionality of $\mathbf{x}_i$) and C (equal to the number of classes), respectively. The number of neurons in the l-th hidden layer is equal to L_l, $l = 1, \ldots, N_L$. Let us denote by $\mathbf{W}_{in} \in \mathbf{R}^{C \times L_1}$ the weights connecting the network's input and first hidden layers and by $\mathbf{W}_{out} \in \mathbb{R}^{L_{N_L} \times C}$ the weights connecting the network's last hidden and output layers. We denote the weights connecting the l-th and $(l+1)$-th hidden layers of the network by $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l+1}}$. For notation simplicity, we assume that the bias values of the network's layers have been absorbed in the network's weights.

The Hierarchical ELM (H-ELM) algorithm [13] is built in an incremental manner by following a process formed by two steps:

1. Unsupervised hierarchical data representation learning and
2. Supervised data projection based on Least Squares regression.

Before unsupervised feature learning, the input vectors $\mathbf{x}_i$ are transformed into an ELM random feature space by using the randomly assigned weights $\mathbf{W}_{in}$. By doing this, each input vector $\mathbf{x}_i$ is mapped to a vector $\mathbf{h}_i^1 \in \mathbb{R}^{L_1}$, where we used the superscript to denote the number of hidden layer defining the feature space used for data representation. We use all vectors $\mathbf{h}_i^1$, $i = 1, \ldots, N$ in order to form the matrix $\mathbf{H}_1 \in \mathbb{R}^{L_1 \times N}$.

Given $\mathbf{H}_l$, we train an AE as follows:

- We determine a random data mapping from $\mathbf{H}_l$ to $\mathbf{H}_{l+1}$ using randomly assigned weights $\tilde{\mathbf{W}}_l \in \mathbb{R}^{L_l \times L_{l+1}}$ followed by a non-linearity.
- We calculate the network's hidden layer weights $\mathbf{W}_l$ by optimizing for:

$$\mathbf{W}_l = \underset{\mathbf{W}_l}{argmin} \|\mathbf{W}_l^T \mathbf{H}_{i+1} - \mathbf{H}_l\|_2^2 + \|\mathbf{W}_l\|_1 \quad (1)$$

  The above-given optimization problem is solved by applying a fast iterative shrinkage-thresholding algorighm (FISTA) [15]. By applying this process, a compact representation of the data representations in the feature space of the network's l-th layer is obtained. This new representation is stacked on top of the existing network (formed by $l+1$ layers).

The above-described process is applied in a hierarchical manner in order to determine the weights of the network's hidden layers and the data representations in the feature space determined by the network's last hidden layer outputs $\mathbf{H}_{N_L-1} \in \mathbb{R}^{N_L \times N}$ are obtained.

After the calculation of $\mathbf{H}_{N_L-1}$, a linear regression problem is solved in order to map $\mathbf{h}_i^{L_N-1}$ to target vectors $\mathbf{t}_i \in \mathbb{R}^C$, $i = 1, \ldots, N$. The elements of the network target vectors $\mathbf{t}_i = [t_{i1}, ..., t_{iC}]^T$ are set to $t_{ik} = 1$ for samples belonging to class k, i.e., when $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$. The network output weights are calculated by solving the following optimization problem:

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

$$\textbf{Subject to:} \quad \mathbf{W}_{out}^T \mathbf{h}_i^{L_N-1} = \mathbf{t}_i - \boldsymbol{\xi}_i, \ i = 1, ..., N, \quad (3)$$

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. The network output weights $\mathbf{W}_{out}$ can be determined by optimizing the dual problem:

$$\begin{aligned} \tilde{\mathcal{J}} &= \frac{1}{2}\|\mathbf{W}_{out}\|_F^2 + \frac{c}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|_2^2 \\ &- \sum_{i=1}^{N}\mathbf{a}_i\left(\mathbf{W}_{out}^T\mathbf{h}_i^{L_N-1} - \mathbf{t}_i + \boldsymbol{\xi}_i\right), \end{aligned} \quad (4)$$

which is equivalent to (2). By calculating the derivatives of $\tilde{\mathcal{J}}$ with respect to $\mathbf{W}_{out}$, $\boldsymbol{\xi}_i$ and $\mathbf{a}_i$ and setting them equal to zero, the network output weights $\mathbf{W}_{out}$ are obtained by:

$$\mathbf{W}_{out} = \left(\mathbf{H}_{N_L-1}\mathbf{H}_{N_L-1}^T + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{H}_{N_L-1}\mathbf{T}^T, \quad (5)$$

or

$$\mathbf{W}_{out} = \mathbf{H}_{N_L-1}\left(\mathbf{H}_{N_L-1}^T\mathbf{H}_{N_L-1} + \frac{1}{c}\mathbf{I}\right)^{-1}\mathbf{T}^T. \quad (6)$$

From the above, it can be seen that for the determination of the network target vectors (and thus for the determination of the network output weights $\mathbf{W}_{out}$), only the labeling information of the training data is exploited. As it has been shown in [14] for single-hidden layer networks, in subspace learning problems this approach provides inferior performance when compared to the adoption of target vectors encoding information concerning both the training data labels and training data geometric properties when represented in the ELM space. In the next section, we formulate a process for the determination of such target vectors for deep networks training.

3. TARGET VECTORS DETERMINATION

Taking into account both c_i and $\mathbf{h}_i^{L_N-1}$, the probability of class k can be expressed using the Bayes formula as:

$$p(k|c_i, \mathbf{h}_i^{L_N-1}) \propto p(c_i, \mathbf{h}_i^{L_N-1}|k)p(k), \tag{7}$$

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 $p(c_i, \mathbf{h}_i^{L_N-1}|k)$ using the training vectors an enormous training set would be required, we can define $p(c_i, \mathbf{h}_i^{L_N-1}|k) = p(\mathbf{h}_i^{L_N-1}|k)p(c_i|k)$, where independence between c_i and $\mathbf{h}_i^{L_N-1}$ is assumed. That is, the probability of class k, given the observations c_i and $\mathbf{h}_i^{L_N-1}$, can be expressed by:

$$p(k|c_i, \mathbf{h}_i^{L_N-1}) \propto p(\mathbf{h}_i^{L_N-1}|k)p(c_i|k)p(k). \tag{8}$$

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, H-ELM assumes that class k can be discriminated from all other classes by using a hyperplane passing through the origin in the high-dimensional ELM space $\mathbb{R}^{N_L}$. This leads to the assumption that class k is homogeneous in $\mathbb{R}^{N_L}$.

Exploiting the class unimodality assumption of ELMs [14], we define the representation of class k in $\mathbb{R}^{N_L}$ by the corresponding class mean vector:

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{j, c_j = k} \mathbf{h}_j^{L_N-1}. \tag{9}$$

Algorithm 1 Supervised Subspace Learning based on Deep Neural networks

Input $\{\mathbf{x}_i, c_i\}$, $i = 1, \ldots, N$, L, N_l, $l = 1, \ldots, N$, $\Phi(\cdot)$.
Randomly assign input parameters $\mathbf{W}_{in} \in \mathbb{R}^{C \times L_1}$.
Perform ELM mapping from $\mathbf{x}_i$ to $\mathbf{h}_i^1$, $i = 1, \ldots, N$.
for $l = 1{:}N_{L-1}$ **do**
  Perform a random ELM mapping using $\tilde{\mathbf{W}}_l \in \mathbb{R}^{L_l \times L_{l+1}}$
  Optimize (1) to determine $\mathbf{W}_l$.
end for
Using $\mathbf{h}_i^{N_{L-1}}$, $i = 1, \ldots, N$ determine target vectors $\mathbf{t}_i$, $i = 1, \ldots, N$ based on (8).
Calculate $\mathbf{W}_{out}$ from (5).
return $\mathbf{W}_{in}, \mathbf{W}_{out}$ and $\tilde{\mathbf{W}}_l$, $l = 1, \ldots, N_{L-1}$.

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 $p(\mathbf{h}_i^{L_N-1}|k)$ can be calculated by:

$$p(\mathbf{h}_i^{L_N-1}|k) = \frac{d(\mathbf{m}_k, \mathbf{h}_i^{L_N-1})}{\sum_{j=1}^{C} d(\mathbf{m}_j, \mathbf{h}_i^{L_N-1})}, \tag{10}$$

where $d(\mathbf{m}_k, \mathbf{h}_i^{L_N-1})$ is a function denoting the similarity of $\mathbf{h}_i^{L_N-1}$ and $\mathbf{m}_k$. In order to increase class discrimination, we employ the following similarity measure in our experiments:

$$d(\mathbf{m}_k, \mathbf{h}_i^{L_N-1}) = exp\left(-\gamma \|\mathbf{m}_k - \mathbf{h}_i^{L_N-1}\|_2^2\right). \tag{11}$$

In (11), $\gamma > 0$ is a parameter that is used in order to scale the Euclidean distance between $\mathbf{m}_k$ and $\mathbf{h}_i^{L_N-1}$. In this paper we set $\gamma = \frac{1}{r\sigma^2}$, where σ is set equal to the mean Euclidean distance between $\mathbf{h}_i^{L_N-1}$, $1 = 1, \ldots, N$ and $\mathbf{m}_k$.

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

$$p(c_i|k) = \frac{d(\mathbf{m}_k, \mathbf{m}_{c_i})}{\sum_{j=1}^{C} d(\mathbf{m}_j, \mathbf{m}_{c_i})}, \tag{12}$$

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

$$d(\mathbf{m}_k, \mathbf{m}_{c_i}) = exp\left(-\delta \|\mathbf{m}_k - \mathbf{m}_{c_i}\|_2^2\right). \tag{13}$$

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. 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 $p(k) = 1$, $k = 1, \ldots, C$.

The proposed method for supervised subspace learning based on deep randomized networks is illustrated in Algorithm 1.

Table 1. UCI datasets details.

Dataset	Samples	Dimensions (D)	Classes (C)
Australian	690	14	2
Column (2c)	310	6	2
Column (3c)	310	6	3
German	1000	24	2
Glass	241	9	6
Heart	270	13	2
Indians	768	8	2
Ionosphere	351	34	2
Madelon	2600	500	2
Relax	182	12	2
Segmentation	2310	19	7
Spect	267	22	2
Spectf	267	44	2
Synth.Cont.	600	60	6
TAE	151	5	3
TicTacToe	958	9	2

Table 2. Performance of the two approaches in UCI datasets.

	$N_L = 250$		$N_L = 500$	
Dataset	H-ELM	Proposed	H-ELM	Proposed
Australian	$64.35(\pm 0.01)$	$\mathbf{64.44(\pm 0.28)}$	$66.27(\pm 0.32)$	$\mathbf{67.27(\pm 0.37)}$
Column (2c)	$77.19(\pm 0.31)$	$\mathbf{77.87(\pm 0.41)}$	$77.48(\pm 0.2)$	$\mathbf{78(\pm 0.82)}$
Column (3c)	$\mathbf{78.97(\pm 0.2)}$	$78.61(\pm 0.31)$	$76.16(\pm 0.1)$	$\mathbf{78.65(\pm 0.2)}$
German	$62.39(\pm 0.35)$	$\mathbf{64.13(\pm 0.54)}$	$63.29(\pm 0.01)$	$\mathbf{64.01(\pm 0.01)}$
Glass	$57.2(\pm 1.05)$	$\mathbf{61.07(\pm 1.22)}$	$60.35(\pm 0.27)$	$\mathbf{62.37(\pm 0.59)}$
Heart	$61.63(\pm 0.47)$	$\mathbf{64.89(\pm 0.25)}$	$63.48(\pm 0.7)$	$\mathbf{63.59(\pm 0.35)}$
Indians	$63.65(\pm 0.33)$	$\mathbf{64.79(\pm 0.21)}$	$62.15(\pm 1.36)$	$\mathbf{63.74(\pm 0.2)}$
Ionosphere	$84.29(\pm 0.98)$	$\mathbf{87.19(\pm 0.9)}$	$81.09(\pm 0.36)$	$\mathbf{83.02(\pm 0.53)}$
Madelon	$57.49(\pm 0.01)$	$\mathbf{58.97(\pm 0.01)}$	$58.07(\pm 0.01)$	$\mathbf{59.97(\pm 0.01)}$
Relax	$65.75(\pm 0.33)$	$\mathbf{66.69(\pm 0.7)}$	$64.6(\pm 0.86)$	$\mathbf{67.18(\pm 0.34)}$
Segmentation	$\mathbf{82.07(\pm 0.01)}$	$78.77(\pm 0.31)$	$78.09(\pm 0.3)$	$\mathbf{80.59(\pm 0.01)}$
Spect	$74.67(\pm 0.71)$	$\mathbf{74.93(\pm 1.19)}$	$74.34(\pm 0.72)$	$\mathbf{74.78(\pm 0.5)}$
Spectf	$73.91(\pm 0.34)$	$\mathbf{76.41(\pm 0.01)}$	$75.33(\pm 0.94)$	$\mathbf{76.33(\pm 0.24)}$
Synth.Cont.	$\mathbf{90.55(\pm 0.69)}$	$89.93(\pm 0.32)$	$92.1(\pm 0.21)$	$\mathbf{93.23(\pm 0.32)}$
TAE	$61.83(\pm 0.62)$	$\mathbf{65.42(\pm 0.42)}$	$64.5(\pm 0.6)$	$\mathbf{64.54(\pm 0.82)}$
TicTacToe	$\mathbf{69(\pm 0.36)}$	$68.66(\pm 0.92)$	$60.65(\pm 1.61)$	$\mathbf{63.44(\pm 0.1)}$

4. EXPERIMENTS

In this Section we provide experiments conducted in order to compare the two approaches. We have employed 16 publicly available datasets coming from the UCI Machine Learning repository [16] to this end. Information related to the datasests used is illustrated in Table 1. For performance evaluation we have applied the five fold cross-validation process where the performance of each algorithm in one experiment is evaluated by calculating the mean classification rate over all folds. On each dataset, we have applied 10 experiments, and we measured the performance of each algorithm by calculating the mean classification rate and the standard deviation over all 10 experiments.

We have tested the performance of two network topologies, each of which is formed by three hidden layers. The number of hidden layer neurons used in these networks is set to $L_l = 250,\ l = 1, \ldots, N_L$ and $L_l = 500,\ l = 1, \ldots, N_L$, respectively. After training the networks, we mapped both the training and test data to the obtained feature space and we applied classification using the Nearest Neighbor classifier. The optimal regularization parameter value has been determined by applying a grid search strategy using the vales $c = 10^{(0:6)}$. Experimental results are illustrated in Table 2.

As can be seen in Table 2, the exploitation of target vectors encoding both the labeling information of the training data and geometric properties of the feature space determined by the network's last hidden layer outputs leads to better performance in twelve (out of sixteen) cases when using a network with $L = 3$ and $N_l = 250,\ l = 1, 2, 3$. When using a larger network topology, ($N_l = 500$) the adoption of the proposed target vectors leads to better performance in all cases. This is interesting, since we expect that the adoption of larger network topologies will lead to reacher data representations and, thus, more discriminant feature spaces for supervised subspace learning.

5. CONCLUSION

In this paper, we proposed a supervised subspace learning method that is based on deep randomized neural networks and regression models. We showed that for efficient subspace learning the exploitation of information related to both the labels of the training data and the geometric properties of the feature space determined by the network's last hidden layer outputs should be taken into account. For this reason, we formulated a Bayesian model for the calculation of appropriate target vectors that can be subsequently used for deep neural network-based supervised subspace learning. Experimental results on publicly available dataset show that the proposed method can outperform the original approach. Possible future research directions would be the exploitation of kernel versions of ELM networks [17, 18] and the exploitation of geometric class/data relationships [9, 19, 20] and discriminant learning criteria [21] for forming deep network architectures.

6. REFERENCES

[1] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is Extreme Learning Machine feasible? a theoretical assessment (Part I)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 7–20, 2015.

[2] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is Extreme Learning Machine feasible? a theoretical assessment (Part II)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 21–34, 2015.

[3] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[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] 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.

[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, 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, 2004. Proceedings*, vol. 2, pp. 985–990, 2004.

[9] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum Class Variance Extreme Learning Machine for human action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 11, pp. 1968–1979, 2013.

[10] A. Iosifidis, A. Tefas, and I. Pitas, "Dynamic action recognition based on Dynemes and Extreme Learning Machine," *Pattern Recognition Letters*, vol. 34, pp. 1890–1898, 2013.

[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] 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.

[13] J. Tang, C. Deng, and G.B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transanctions on Neural Networks and Learning Systems, D.O.I.: 10.1109/TNNLS.2015.2424995*, 2015.

[14] A. Iosifidis, "Extreme learning machine based supervised subspace learning," *Neurocomputing*, vol. 167, pp. 158–164, 2015.

[15] A. Beck and M. Tebboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1.

[16] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

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

[18] A. Iosifidis and M. Gabbouj, "On the kernel extreme learning machine speedup," *Pattern Recognition Letters*, vol. 68, pp. 205–210, 2015.

[19] A. Iosifidis, A. Tefas, and I. Pitas, "Minimum variance extreme learning machine for human action recognition," *International Conference on Acoustics, Speech and Signal Processing*, 2014.

[20] A. Iosifidis, A. Tefas, and I. Pitas, "exploiting local class information in extreme learning machine," *International Conference on Neural Computation Theory and Applications*, 2014.

[21] A. Iosifidis, A. Tefas, and I. Pitas, "Graph Embedded Extreme Learning Machine," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 311–324, 2016.