ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

Kernel-Based Multilayer Extreme Learning Machines for Representation Learning

Vasimkha Taslimkha

Department of Computer Engineering, S.No, 39, Narhe Gaon Rd, Narhe, Pune, Maharashtra, India

ABSTRACT: Extreme learning machine (ELM) is an emerging learning algorithm for the generalized single hidden layer feedforward neural networks, of which the hidden node parameters are randomly generated and the output weights are analytically computed. However, due to its shallow architecture, feature learning using ELM may not be effective for natural signals (e.g., images/videos), even with a large number of hidden nodes. Recently, multilayer extreme learning machine (ML-ELM) was applied to stacked autoencoder (SAE) for representation learning. In contrast to traditional SAE, the training time of ML-ELM is significantly reduced from hours to seconds with high accuracy. However, ML-ELM suffers from several drawbacks: 1) manual tuning on the number of hidden nodes in every layer is an uncertain factor to training time and generalization; 2) random projection of input weights and bias in every layer of ML-ELM leads to suboptimal model generalization; 3) the pseudoinverse solution for output weights in every layer incurs relatively large reconstruction error; and 4) the storage and execution time for transformation matrices in representation learning are proportional to the number of hidden layers. Inspired by kernel learning, a kernel version of ML-ELM is developed, namely, multilayer kernel ELM (ML-KELM), whose contributions are: 1) elimination of manual tuning on the number of hidden nodes in every layer; 2) no random projection mechanism so as to obtain optimal model generalization; 3) exact inverse solution for output weights is guaranteed under invertible kernel matrix, resulting to smaller reconstruction error; and 4) all transformation matrices are unified into two matrices only, so that storage can be reduced and may shorten model execution time. Benchmark data sets of different sizes have been employed for the evaluation of ML-KELM. Experimental results have verified the contributions of the proposed ML-KELM. The improvement in accuracy over benchmark data sets is up to 7%

KEYWORDS: Deep learning (DL), deep neural network (DNN), extreme learning machine (ELM), multilayer perceptron (MLP), random feature mapping, Kernel learning, multilayer extreme learning machine (ML-ELM), representation learning, stacked autoencoder (SAE).

I. INTRODUCTION

During the past years, extreme learning machine (ELM) has been becoming an increasingly significant research topic for machine learning and artificial intelligence, due to its unique characteristics, i.e., extremely fast training, good generalization, and universal approximation/ lassification capability. ELM is an effective solution for the single hidden layer feedforward networks (SLFNs), and has been demonstrated to have excellent learning accuracy/speed in various applications, such as face classification, image segmentation , and human action recognition.

Unlike the other traditional learning algorithms, e.g., back propagation (BP)-based neural networks (NNs), or support vector machine (SVM), the parameters of hidden layers of the ELM are randomly generated and need not to be tuned, thus the hidden nodes could be established before the training samples are acquired. Theoretically, Huang et al. have proved that the SLFNs with randomly generated hidden neurons and the output weights tuned by regularized least square maintain its universal approximation capability, even without updating the parameters of hidden layers. In addition, solving the regularized least squares problem in ELM is faster than that of the quadratic programming problem in standard SVM or gradient method in traditional BP-based NNs. Thus, ELM tends to achieve faster and better generalization performance than those of NNs and SVM.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

Representation learning is a kind of methods that automatically extracts an effective representation from a set of available data (x, t), whose attractive advantage is the elimination of burdens of human intervention and expert knowledge. Out of different representation learning methods, autoencoder (AE) is a direct and efficient computational algorithm, which learns the data representation by replacing output t with input x itself. By stacking multiple AEs into a stacked AE (SAE), more expressive and complex data representation can be learned in a hierarchical manner. There is a wide range of applications for SAE, including dimension reduction and transfer learning. For dimension reduction, multimedia data are a good example whose high-dimensional raw input is usually noisy and unsuitable for direct processing. SAE takes the raw multimedia data, and learns a reduced yet informative representation by setting the number of hidden nodes L_i in the ith hidden layer fewer than $L_{(i-1)}$ in the $(i-1)$th hidden layer.

For transfer learning the effective representation in source tasks is transferred to the target task, so that the performance of target task can be improved. For example, in speech emotion recognition, only a few corpora are available while they are highly dissimilar in terms of spoken language. SAE learns a representation trained on a specific emotion class from target data set. Then, this representation is applied to the source data set with the same emotion class in order to reconstruct the target data set for emotion classification. These examples illustrate the significance of SAE. However, the iterative layerwise training process by back-propagation in SAE is extremely time-consuming. Hence, a time-efficient improvement over SAE is desired.

Recently, multilayer extreme learning machine (ML-ELM) was proposed by employing ELM-based AE (ELM-AE) in SAE. The major contribution of ML-ELM is its extremely fast training speed, aiming to resolve the time-consuming issue of deep learning. From the experimental results, ML-ELM achieves much faster speed with even higher generalization than SAE, deep belief network, deep Boltzmann machine, and so on. The ELM mechanism is based on random projection, where the input weights a and bias b for the input layer of AE are randomly generated while the weights for output layer (i.e., output weights) can be analytically calculated without any iteration. Another advantage of ML-ELM is that representation learning and classification can be integrated into a single learning process where multiple layers of ELM-AE are for representation learning, followed by a final layer of ELM for classification.

II. LITERATURE SURVEY

A. *Extreme Learning Machine for Multilayer Perceptron*

Extreme learning machine (ELM) is an emerging learning algorithm for the generalized single hidden layer feedforward neural networks, of which the hidden node parameters are randomly generated and the output weights are analytically computed. However, due to its shallow architecture, feature learning using ELM may not be effective for natural signals (e.g., images/videos), even with a large number of hidden nodes. To address this issue, in this paper, a new ELM-based hierarchical learning framework is proposed for multilayer perceptron. The proposed architecture is divided into two main components:

1) self-taught feature extraction followed by supervised feature classification and 2) they are bridged by random initialized hidden weights. The novelties of this paper are as follows:
1) unsupervised multilayer encoding is conducted for feature extraction, and an ELM-based sparse autoencoder is developed via ***l*1** constraint. By doing so, it achieves more compact and meaningful feature representations than the original ELM; 2) by exploiting the advantages of ELM random feature mapping, the hierarchically encoded outputs are randomly projected before final decision making, which leads to a better generalization with faster learning speed; and 3) unlike the greedy layerwise trainin of deep learning (DL), the hidden layers of the proposed framework are trained in a forward manner. Once the previous layer is established, the weights of the current layer are fixed without fine-tuning. Therefore, it has much better learning efficiency than the DL. Extensive experiments on various widely used classification data sets show that the proposed algorithm achieves better and faster convergence than the existing state-of-the-art hierarchical learning methods. Furthermore, multiple applications in computer vision further confirm the generality and capability of the proposed learning scheme.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

B. An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels

Extreme learning machines (ELMs) basically give answers to two fundamental learning problems:

(1) Can fundamentals of learning (i.e., feature learning, clustering, regression and classification) be made without tuning hidden neurons (including biological neurons) even when the output shapes and function modeling of these neurons are unknown?

(2) Does there exist unified framework for feedforward neural networks and feature space methods? ELMs that have built some tangible links between machine learning techniques and biological learning mechanisms have recently attracted increasing attention of researchers in widespread research areas.

This paper provides an insight into ELMs in three aspects, viz:

random neurons, random features and kernels. This paper also shows that in theory ELMs (with the same kernels)

tend to outperform support vector machine and its variants in both regression and classification applications with much easier implementation.

C. Random Feature Mapping with Signed Circulant Matrix Projection

Random feature mappings have been successfully used for approximating non-linear kernels to scale up kernel methods. Some work aims at speeding up the feature mappings, but brings increasing variance of the approximation. In this paper, we propose a novel random feature mapping method that

uses a signed Circulant Random Matrix (CRM) instead of an unstructured random matrix to project input data. The signed CRM has linear space complexity as the whole signed CRM can be recovered from one column of the CRM, and ensures loglinear time complexity to compute the feature

mapping using the Fast Fourier Transform (FFT). Theoretically, we prove that approximating Gaussian kernel using our mapping method is unbiased and does not increase the variance. Experimentally, we demonstrate that our proposed mapping method is time and space efficient while retaining similar accuracies with state-of-the-art random feature mapping methods. Our proposed random feature mapping method can be implemented easily and make kernel methods scalable and practical for large scale training and predicting problems.

III. PROPOSED METHODOLOGY

The proposed ML-KELM is developed by stacking multiple kernel version of ELM-AEs, namely, KELM-AE. In this section, the details of KELM-AE and ML-KELM are discussed.

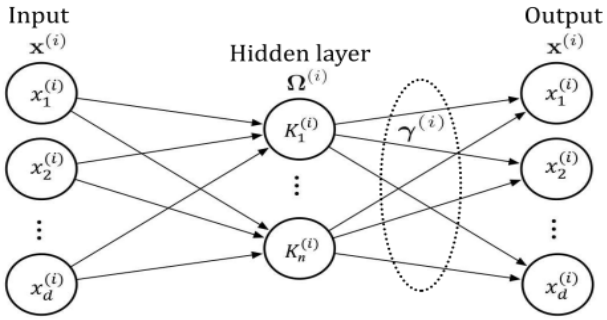

Fig. Architecture of the ith KELM-AE, in which the hidden layer

A. KELM-AE

Similar to ELM-AE, KELM-AE learns the data transformation from hidden layer to output layer. As shown in Fig., the input matrix $\mathbf{X^{(i)}}$ is mapped into a kernel matrix $\mathbf{\Omega^{(i)}}$ via kernel function $\mathbf{K^{(i)}}$ $(x_k , xj) = \exp(- \| x_k - x_j \| / 2\sigma_i^2)$ (i.e.,

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

RBF with parameter σ_i in our case). In order to avoid confusion, $\tilde{\Gamma}$ is used to represent the ith transformation matrix in KELM-AE, which can be learned similar to ELM-AE in $\tilde{\Gamma}$

$$\Omega^{(i)} \tilde{\Gamma}^{(i)} = X^{(i)}$$

$\tilde{\Gamma}^{(i)}$ is calculated by

$$\tilde{\Gamma}^{(i)} = (I/C + \Omega^{(i)})^{-1} X^{(i)}$$

In the final step of transformation procedure, the data representation X^{i+1} is obtained similar to

$$X^{i+1} = g(X^{(i)}(\tilde{\Gamma}^{(i)})^T)$$

where g is an arbitrary activation function. If the ith layer has the same dimension as the (i + 1)th layer, g can be chosen as linear piecewise. In ML-KELM, all $\tilde{\Gamma}^{(i)}$ (i>1) are of the same dimension n × n (i.e., square matrix). Then, linear piecewise activation function can be applied to all $\tilde{\Gamma}^{(i)}$, except $\tilde{\Gamma}^{(1)}$.

Hence, the transformation matrices $\tilde{\Gamma}^{(i)}$ for i > 1 can be unified into a single matrix $\tilde{\Gamma}_{unified}$ the representation capability. Consequently, only two matrices $\tilde{\Gamma}^{(1)}$ and $\tilde{\Gamma}_{unified}$ can represent all transformations. KELM-AE is detailed in Algorithm 1.

ML-KELM adopts two separate learning procedures as in H-ELM so that ML-KELM can also maintain the universal approximation of ELM. In the stage of representation learning, each pair of $\tilde{\Gamma}^{(i)}$ and $X^{(i)}$ (for ith KELM-AE) can be calculated. Finally, the final data representation X final is calculated and fed as input to train a K-ELM classification model as follows:

$$\Omega^{(final)} \beta = T$$

where $\Omega^{(final)}$ is the kernel matrix obtained from $X^{(final)}$. The output weight β is calculated by

$$\beta = (I/C + \Omega^{(final))-1} T$$

Similarly, ML-KELM does not perform iterative fine-tuning once all the parameters are fixed in every layer. On the other hand, the proposed ML-KELM does not need to tune L_i, a_i, and b_i for any layer, which is a significant appeal upon multilayer neural networks, such as ML-ELM and H-ELM. The procedure of ML-KELM is detailed in Algorithm 2.

Algorithm 1 KELM-AE for the ith Layer

Input: Input matrix $\mathbf{X}^{(i)}$, regularization C_i, kernel parameter σ_i, and activation function g_i

Output: New data representation $\mathbf{X}^{(i+1)}$ and transformation matrix $\tilde{\mathbf{\Gamma}}^{(i)}$

Step 1: Calculate the kernel matrix $\Omega^{(i)}_{k,j} \leftarrow K(\mathbf{x}_k, \mathbf{x}_j, \sigma_i)$

Step 2: Calculate the transformation matrix $\tilde{\Gamma}^{(i)} \leftarrow (\frac{\mathbf{I}}{C_i} + \Omega^{(i)})^{-1}\mathbf{X}^{(i)}$

Step 3: Calculate new data representation $\mathbf{X}^{(i+1)} \leftarrow g_i(\mathbf{X}^{(i)}(\tilde{\Gamma}^{(i)})^T)$

return $\mathbf{X}^{(i+1)}$, and $\tilde{\Gamma}^{(i)}$

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

IV. RESULT AND DISCUSSIONS

Algorithm 2 Proposed ML-KELM

Input: input matrix $\mathbf{X}^{(1)}$, output matrix $\mathbf{T}$, regularization C_i for all layers, kernel parameters of σ_i for all layers, the number of layers N_{Layer}, and activation g_i

Output: representation matrix $\mathbf{X}^{\text{final}}$, two transformation matrices $\tilde{\mathbf{\Gamma}}^{(1)}$ and $\tilde{\mathbf{\Gamma}}_{\text{unified}}$, and output weight $\boldsymbol{\beta}$

Step 1: Calculate $\mathbf{X}^{(2)}, \tilde{\mathbf{\Gamma}}^{(1)} \leftarrow KELMAE(\mathbf{X}^{(1)}, C_1, \sigma_1, g_1)$

Step 2: Calculate $\mathbf{X}^{(3)}, \tilde{\mathbf{\Gamma}}^{(2)} \leftarrow KELMAE(\mathbf{X}^{(2)}, C_2, \sigma_2, g_2)$

Step 3: Initialize $\tilde{\mathbf{\Gamma}}_{\text{unified}} \leftarrow \tilde{\mathbf{\Gamma}}^{(2)}$

for $i = 3 : N_{Layer} - 1$ **do**

Step 4: Calculate $\mathbf{X}^{(i+1)}, \tilde{\mathbf{\Gamma}}^{(i)} \leftarrow KELM\text{-}AE(\mathbf{X}^{(i)}, C_i, \sigma_i, g_i)$

Step 5: Update $\tilde{\mathbf{\Gamma}}_{\text{unified}} \leftarrow \tilde{\mathbf{\Gamma}}^{(i)} \tilde{\mathbf{\Gamma}}_{\text{unified}}$

Step 6: $i \leftarrow N_{Layer}$; $\mathbf{X}^{\text{final}} \leftarrow \mathbf{X}^{(i)}$

Step 7: Calculate $\mathbf{\Omega}_{k,j}^{(i)} \leftarrow K^{(i)}(\mathbf{x}_k^{(i)}, \mathbf{x}_j^{(i)}, \sigma_i)$

Step 8: Calculate the output weight $\boldsymbol{\beta} \leftarrow (\frac{\mathbf{I}}{C_i} + \mathbf{\Omega}^{(i)})^{-1}\mathbf{T}$

return $\mathbf{X}^{\text{final}}$, $\tilde{\mathbf{\Gamma}}^{(1)}$, $\tilde{\mathbf{\Gamma}}_{\text{unified}}$, and $\boldsymbol{\beta}$

In this section, ML-KELM is evaluated over 20 publicly available benchmark data sets from UCI repository and http://openml.org. All data sets are described in Table I and categorized into small, medium, and large in terms of size in order to have a thorough evaluation on the training time and testing accuracy.

A. Experiment Setup

The experiments are conducted using python 3.6 running on a 3.6-GHz i7 CPU with 16-GB RAM. The number of hidden layers is set to 3 for all experiments. For ML-ELM and H-ELM, the numbers of hidden nodes L_i (i = 1 to 3) are, respectively, set as 100 × m {m =, 2, . . . , 15}.

For the proposed ML-KELM, tuning of L_i is not necessary and RBF kernel is chosen as the kernel function for invertible kernel matrix Ω. For all methods, the regularization parameter C_i (i = 1 to 3) in each layer is, respectively, set as 10 x , {x = −7, −5, . . . , 7}.

For ML-KELM, the kernel parameter σ_i (i = 1 to 3) in each layer is, respectively, set as 10 x {x = −7, −5, . . . , 7}. Tenfold cross validation is used for all experiments for a fair comparison.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

TABLE I
PROPERTIES OF BENCHMARK DATA SETS

Category	Dataset	#Instances	Features
Small	*Bupa*	250	6
	Pima	500	8
	Cmc	1000	9
	Yeast	1000	8
	Segment	1700	19
	Mfeat	1400	649
Medium	*Gina*	2500	970
	Prior	2500	785
	Advertisement	2500	1558
	Isolet	2500	617
	Abalone	3000	8
	Letter	3000	16
	Har 1	3000	562
	Adult 1	3000	122
	Spambase	3500	57
	Waveform	4000	21
Large	*Vowels*	7000	14
	Har 2	7352	562
	Sylva	10000	216
	Adult 2	15000	122

TABLE III
TRAINING TIME (IN SECONDS) FOR ALL DATA SETS

	Dataset	ML-ELM	ML-KELM	H-ELM
Small	*Bupa*	0.02	**0.01**	0.49
	Pima	0.19	**0.03**	0.04
	Cmc	0.15	**0.12**	0.41
	Yeast	0.32	**0.09**	0.28
	Segment	**0.33**	0.6	0.43
	Mfeat	0.53	**0.33**	1.32
Medium	*Gina*	**1.78**	2.28	1.92
	Prior	0.73	1.42	**0.32**
	Advertisement	2.23	**1.99**	3.37
	Isolet	1.71	**1.51**	2.09
	Abalone	1.09	2.10	**0.28**
	Letter	1.48	2.05	**0.57**
	Har 1	**1.62**	2.39	2.10
	Adult 1	**1.39**	2.35	3.3
	Spambase	**2.03**	4.14	2.55
	Waveform	**3.09**	5.36	3.31
Large	*Vowels*	**3.69**	19.26	5.06
	Har 2	**2.39**	24.83	3.10
	Sylva	1.35	42.13	**1.05**
	Adult 2	**1.42**	187.38	4.19

B. Evaluation of Testing Accuracy

For each of ML-ELM, H-ELM, and ML-KELM, the testing accuracy under the best combination of parameters is shown and compared in Table II, where ML-KELM outperforms the other two methods for all data sets (up to 7% for Pima). When there is plenty of data, ML-ELM performs similar to ML-KELM (Table II Large). However, ML-ELM and H-ELM are suboptimal, because the input weight a_i and bias b_i in every layer are randomly generated. In some cases, poorly generated a_i and b_i deteriorate the generalization of ML-ELM and H-ELM (shown in Table II). Therefore, ML-ELM and H-ELM require numerous retrainings under fixed parameters in order to find out the best combination of a_i and b_i for high model generalization. Furthermore, the accuracies of ML-ELM and H-ELM are very sensitive to L_i . In ML-KELM, these two issues have been completely resolved due to the advantages of kernel learning.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

TABLE II
MEAN AND STANDARD DEVIATION OF TESTING ACCURACY (%) OF TENFOLD CROSS VALIDATION UNDER BEST COMBINATION OF PARAMETERS

	Dataset	ML-ELM	ML-KELM	H-ELM
Small	*Bupa*	73.40±6.8	**75.16±6.9**	73.21±7.1
	Pima	74.03±4.6	**79.61±5.1**	73.57±3.1
	Cmc	56.72±3.9	**57.89±4.1**	56.37±1.9
	Yeast	60.12±2.8	**61.42±3.1**	59.85±3.9
	Segment	91.70±0.9	**93.29±1.2**	88.91±2.3
	Mfeat	94.88±1.1	**96.02±1.3**	89.39±3.7
Medium	*Gina*	88.56±0.9	**93.26±1.0**	91.09±1.3
	Prior	94.24±0.5	**95.74±1.7**	94.60±1.3
	Advertisement	93.48±0.8	**95.15±0.7**	92.72±1.5
	Isolet	92.56±0.6	**94.51±1.0**	91.24±1.2
	Abalone	56.28±1.8	**58.94±1.3**	57.80±1.2
	Letter	86.71±1.5	**88.40±1.4**	85.23±1.5
	Har 1	97.92±0.5	**98.53±0.4**	93.95±0.7
	Adult 1	84.18±0.8	**84.33±1.2**	83.20±1.5
	Spambase	83.32±1.3	**86.93±2.1**	76.22±1.2
	Waveform	84.93±1.2	**87.03±0.4**	84.87±0.4
Large	*Vowels*	91.31±1.1	**93.92±0.3**	77.33±1.3
	Har 2	98.46±0.3	**99.22±0.2**	93.93±0.7
	Sylva	97.32±0.1	**98.77±0.1**	96.84±0.3
	Adult 2	84.56±0.0	**85.01±0.0**	84.71±0.0
	Average	84.23±1.6	**86.16±1.6**	82.75±1.8

C. Evaluation of Average Training Time

The average training times of ML-ELM, H-ELM, and ML-KELM are compared and shown in Table III. ML-KELM is the fastest for most of the small data sets. The reason is that the training time of ML-KELM depends on the number of training data n, while ML-ELM and H-ELM depend on the number of hidden nodes L_i . Subject to data complexity, large L_i tends to provide higher model generalization but increases training time. For small data sets, L_i is usually larger than n, so that ML-KELM is mostly with faster training time than ML-ELM and H-ELM. However, ML-KELM becomes slower than ML-ELM and H-ELM in large data sets because of large $n > L_i$. Nevertheless, in ML-ELM and H-ELM, L_i can easily become more than 10 000 for highly nonlinear applications (in our experiments, L i is set to at most 1500, because the data are not highly nonlinear).

Although the average training time of ML-KELM becomes slower for large data sets, it is not the case practically. Note that Table III shows only the average training time for one model with fixed parameters. Although ML-ELM or H-ELM take shorter average training time, they need to thoroughly try numerous combinations of parameters (such as L_i , a_i , and b_i) for the best model, so that ML-ELM or H-ELM may take even longer total training time. Even in the case that thorough combinations of parameters are tried, ML-ELM or H-ELM may not have better model generalization than ML-KELM, due to the issue of accumulated reconstruction error (as shown in Table II). Therefore, it is more appealing to have a multilayer neural network with less user intervention, such as ML-KELM.

D. Evaluation on Highly Nonlinear Application

To further illustrate the advantage of ML-KELM, a highly nonlinear data set Madelon was employed that consists of 2000 training data of 500 features. Madelon was employed in NIPS 2003 feature selection challenge . In highly nonlinear application, ML-ELM requires a large L_i in order to obtain better accuracy. In this experiment, L_i is set to at

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

most 10 000 and the best combination of other parameters (e.g., network structure, regularization, scaling factors, kernel parameter, and so on) is experimentally determined for both ML-ELM and ML-KELM. The network structure of ML-ELM is (500-7000-2500-2) with regularization C and scaling factors (r ho, sig, and sig1) = 10 5 , 0.05, 0.7, 0.8, respectively, for layer 500-7000; 10, 0.05, 0.8, 0.9, respectively, for layer 7000-2500, and 10, 0.05, respectively, for layer 2500-2. On the other hand, for n = 2000, ML-KELM has an automatically determined structure (500-2000-2000-2) with regularization C and kernel parameter σ set to 10 −7 , 10 −7 , respectively, for layer 500-2000; 10 3 , 10 5 , respectively, for layer 2000-2000, and 10 3 , 10 7 , respectively, for layer 2000-2. As shown in Table IV, ML-KELM significantly outperforms ML-ELM in terms of total training time, testing accuracy, and execution time. With kernel learning, ML-KELM can produce significant improvement on highly nonlinear applications.

TABLE IV
AVERAGE PERFORMANCE OF ML-ELM AND ML-KELM FOR *Madelon* UNDER BEST COMBINATION OF PARAMETERS

	Training time (s)	Testing accuracy (%)	Execution time (s)
ML-ELM	8.15	68.72±0.1	0.09
ML-KELM	**0.864**	**72.67±0.1**	**0.06**

V. CONCLUSIONS

In this brief, the proposed ML-KELM is a kernel version of ML-ELM by stacking multiple KELM-AEs, which has resolved several drawbacks over ML-ELM and H-ELM.

1) ML-KELM does not need to tune the parameters (L_i , a_i , and b_i) for all layers as in ML-ELM and H-ELM.

2) ML-KELM learns an optimal model in one-shot under fixed parameters.

3) The transformation matrices $\tilde{\mathbf{\Gamma}}^{(i)}$ can be learned via exact inverse of kernel matrix rather than pseudoinverse, resulting to better reconstruction of data representation and better model generalization. In the experiments, testing accuracy is improved up to 7% (for Pima).

4) Transformation matrices $\tilde{\mathbf{\Gamma}}^{(i)}$ for all layers can be combined into two transformation matrices only. Their storage sizes are compact and fixed regardless to the number of hidden layers, which are helpful to those memory-critical systems. In addition, model execution time can be much shortened benefited from the unified transformation matrices.I

n a nutshell, ML-KELM has a remarkable improvement on the learning of data representation over ML-ELM and H-ELM with less user intervention.

REFERENCES

[1] Y. Bengio, “Learning deep architectures for AI,” Found. Trends Mach. Learn., vol. 2, no. 1, pp. 1–127, 2009.
[2] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.
[3] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik, “Dimensionality reduction: A comparative review,” J. Mach. Learn. Res., vol. 10, nos. 1–41, pp. 66–71, Oct. 2009.
[4] W. Wang, Y. Huang, Y. Wang, and L. Wang, “Generalized autoencoder: A neural network framework for dimensionality reduction,” in Proc.IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2014,pp. 490–497.
[5] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, “Sparse autoencoder-based feature transfer learning for speech emotion recognition,” in Proc. Humaine Assoc. Conf. Affective Comput. Intell. Interaction (ACII), Sep. 2013, pp. 511–516.
[6] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,”Unsupervised Transf. Learn. Challenges Mach. Learn., vol. 7, no. 1, pp. 37–50, 2012.
[7] C.-K. Shie, C.-H. Chuang, C.-N. Chou, M.-H. Wu, and E. Y. Chang, “Transfer representation learning for medical image analysis,” in Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Aug. 2015, pp. 711–714.
[8] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, “Representational learning with extreme learning machine for big data,” IEEE Intell. Syst., vol. 28, no. 6, pp. 31–34, Nov. 2013.
[9] C. L. Lekamalage et al., “Dimension reduction with extreme learning machine,” IEEE Trans. Image Process., vol. 25, no. 8, pp. 3906–3918, Aug. 2016.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 8, Issue 6, June 2019

[10] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2. Jul. 2004, pp. 985–990.
[11] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," IEEE Trans. Neural Netw. Learn. Syst., vol. 27, no. 4, pp. 809–821, Apr. 2016.
[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing, vol. 70, nos. 1–3, pp. 489–501, 2006.
[13] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "Online sequential extreme learning machine with kernels," IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 9, pp. 2214–2220, Sep. 2015.
[14] 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, Apr. 2012.
[15] I. Steinwart and A. Christmann, Support Vector Machines. New York, NY, USA: Springer, 2008.
[16] T. Greville, "The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations," SIAM Rev., vol. 1, no. 1, pp. 38–43, 1959.
[17] K. He, X. Zhang, S. Ren, and J. Sun. (2015). Deep Residual Learning for Image Recognition. [Online]. Available: http://www.cvfoundation.org/openaccess/content_cvpr_2016/papers/He_Dee%p_Residual_Learning_CVPR_2016_paper.pdf
[18] C. Yan et al., "Efficient parallel framework for HEVC motion estimation on many-core processors," IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 12, pp. 2077–2089, Dec. 2014.
[19] C. Yan, Y. Zhang, F. Dai, J. Zhang, L. Li, and Q. Dai, "Efficient parallel HEVC intra-prediction on many-core processor," Electron. Lett., vol. 50, no. 11, pp. 805–806, 2014.
[20] C. Yan, Y. Zhang, F. Dai, X. Wang, L. Li, and Q. Dai, "Parallel deblocking filter for HEVC on many-core processor," Electron. Lett., vol. 50, no. 5, pp. 367–368, Feb. 2014.
[21] M. Lichman, "UCI machine learning repository," School Inf. Comput.Sci., Univ. California, Irvine, CA, USA, Tech. Rep., 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[22] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "OPENML: Networked science in machine learning," ACM SIGKDD Explorations Newslett., vol. 15, no. 2, pp. 49–60, 2014.
[23] B. Scholkopf and A. J. Smola, Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2001.
[24] C. A. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," Constructive Approx., vol. 2, no. 1, pp. 11–22, 1986.
[25] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in Proc. Adv. Neural Inf. Process. Syst., 2004, pp. 545–552.