Dimension Reduction With Extreme Learning Machine

Liyanaarachchi Lekamalage Chamara Kasun, Yan Yang, Guang-Bin Huang, *Senior Member, IEEE*, and Zhengyou Zhang, *Fellow, IEEE*

***Abstract*— Data may often contain noise or irrelevant information, which negatively affect the generalization capability of machine learning algorithms. The objective of dimension reduction algorithms, such as principal component analysis (PCA), non-negative matrix factorization (NMF), random projection (RP), and auto-encoder (AE), is to reduce the noise or irrelevant information of the data. The features of PCA (eigenvectors) and linear AE are not able to represent data as parts (e.g. nose in a face image). On the other hand, NMF and non-linear AE are maimed by slow learning speed and RP only represents a subspace of original data. This paper introduces a dimension reduction framework which to some extend represents data as parts, has fast learning speed, and learns the between-class scatter subspace. To this end, this paper investigates a linear and non-linear dimension reduction framework referred to as extreme learning machine AE (ELM-AE) and sparse ELM-AE (SELM-AE). In contrast to tied weight AE, the hidden neurons in ELM-AE and SELM-AE need not be tuned, and their parameters (e.g, input weights in additive neurons) are initialized using orthogonal and sparse random weights, respectively. Experimental results on USPS handwritten digit recognition data set, CIFAR-10 object recognition, and NORB object recognition data set show the efficacy of linear and non-linear ELM-AE and SELM-AE in terms of discriminative capability, sparsity, training time, and normalized mean square error.**

***Index Terms*— Extreme learning machine (ELM), principal component analysis (PCA), non-negative matrix factorization (NMF), random projection (RP), auto-encoder (AE), dimension reduction.**

I. Introduction

MACHINE learning algorithms aim to learn potential functions which map input data to target outputs. However, when input data consist of redundant features or noise, the generalization capability of machine learning methods will be negatively affected, and thus, feature extraction and dimension reduction such as Principal Component Analysis (PCA) [1], [2], Non-negative Matrix Factorization (NMF) [3], Random Projection (RP) [4], [5] and auto-encoder (AE) [6]–[18] are often required to preprocess training data in order to achieve better generalization performance.

PCA is a holistic-based representation algorithm which represents the original data sample to different degrees, and learn a set of linearly uncorrelated features named principal components that describe the variance of data. Dimension reduction is achieved by projecting the input data via a subset of principal components that describes the most variance of the data. The features that have less contribution to the variances are considered to be less descriptive and therefore removed. PCA algorithm converges to a global minimum.

NMF is a parts-based representation algorithm which decomposes the data to a positive basis matrix and positive coefficient matrix. Positive basis matrix learns the commonly occurring parts in data (e.g. nose in a face image) and the positive coefficient matrix indicates the degree in which the commonly occurring parts reconstruct the data. NMF achieves dimension reduction by projecting the data along the positive basis matrix retaining the commonly occurring parts in the data while removing rarely occurring parts in the data.[1]

RP is a computationally efficient method for dimension reduction which retains the Euclidean distance between data points in the reduced dimension space. RP is achieved by projecting the data along an orthogonal random matrix [4] or sparse random matrix [5].

AE is a neural network based dimension reduction algorithm which has output data equal to input data. If the input layer neurons are larger than the hidden layer neurons AE perform dimension reduction. Linear AE learns the variance information similar to PCA [6], and non-linear AE learns non-linear features [7]–[10], [13]–[18]. Tied weight auto-encoder (TAE) is an AE which has the same weights for the input layer and output layer [13].

Extreme Learning Machine (ELM) [19]–[25] was originally proposed to train "generalized" single-hidden layer feedforward neural networks (SLFNs) with fast learning speed, good generalization capability and provides a unified learning paradigm for regression and classification [24]. Furthermore, ELM is efficient in hierarchal learning [26]–[28]. The essence of

Manuscript received May 4, 2015; revised January 28, 2016 and April 14, 2016; accepted May 9, 2016. Date of publication May 18, 2016; date of current version June 28, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christos Bouganis.

L. L. C. Kasun and G.-B. Huang are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: chamarak001@ntu.edu.sg; egbhuangg@ntu.edu.sg).

Y. Yang was with the Energy Research Institute, Nanyang Technological University, Singapore 639798. He is now with the Center of Intelligent Acoustics and Immersive Communications, School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China, and also with the School of Information Science and Technology, Southeast University, Nanjing 210096, China (e-mail: y.yang@nwpu.edu.cn).

Z. Zhang is with Microsoft Corporation, Redmond, WA 98052-6399 USA (e-mail: zhang@microsoft.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2016.2570569

[1]For example, in a dataset which consists of humans the commonly occurring parts will be legs, hands, torso and faces while briefcase will be a rarely occurring part in the image.

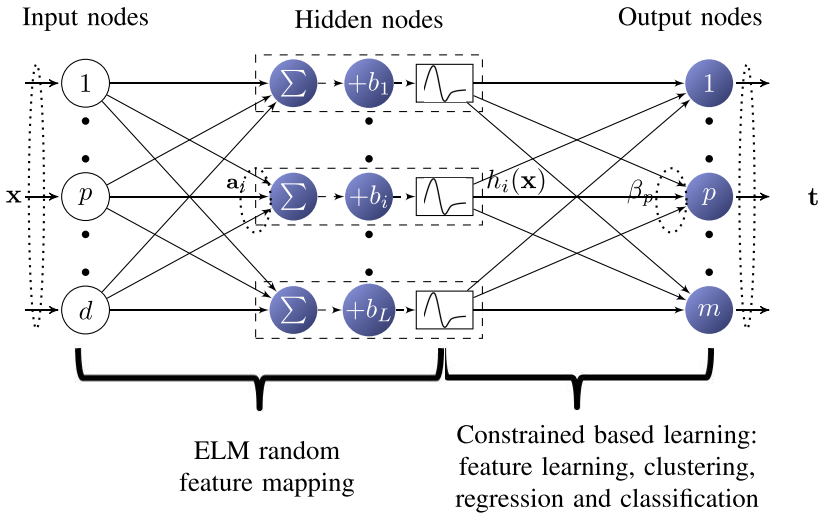


Fig. 1. ELM network structure: The hidden node parameters $(\mathbf{a}_i, b_i)$ are randomly generated. $h_i(\mathbf{x}) = g(\mathbf{a}_i, \mathbf{x}, b_i)$ is the output of i^{th} hidden node for input $\mathbf{x}$ where g is a non-linear piecewise continuous function. This figure shows a specific additive type neuron structure.

ELM is that the hidden node parameters can be assigned randomly independent of training data and need not be tuned, and the output weights are adjusted based on application dependent constraints. Its concrete biological evidence has been found recently [25], [29]–[33]. In contrast to the commonly used Back Propagation (BP) algorithm [34] which minimizes only the training error, ELM minimizes both the training error and the norm of the output weights. According to Bartlett's theory [35], minimizing the norm of the weights leads to better generalization performance. ELM has been widely used in regression and classification problems due to its wide variety of feature mapping functions (sigmoid, hard-limit, Gaussian, multi-quadratic, wavelet, Fourier series, etc) and its ability of handling both large and small datasets efficiently.

The outputs of ELM hidden nodes are referred to as the ELM feature mapping and can be used for dimension reduction (as shown in Figure 1). The commonly referred RP is a special case of ELM feature mapping [25]. In contrast to RP, ELM has a bias and non-linear activation function. Furthermore, ELM can approximate any continuous target function [19]–[25].

When the number of hidden layer neurons in an AE is smaller than that of input neurons it learns to reduce the dimension of data [36]. Motivated by the fact that AE could be used for dimension reduction, this paper investigates AE with random neurons in terms of: 1) theoretical investigation of the proposed linear AE; 2) Normalized Mean Square Error (NMSE); 3) discriminative capability of the features; 4) sparsity of the features and training time. This paper investigates linear and non-linear Extreme Learning Machine Auto-Encoder (ELM-AE) [36] and Sparse Extreme Learning Machine Auto-Encoder (SELM-AE) that has orthogonal and sparse random input neurons which are not tuned. The main contributions of the paper is as follows:

1) In contrast to the perception that linear AE learns the variance information, the proposed linear ELM-AE and linear SELM-AE learn the between-class scatter matrix which reduce the distance of data points belonging to the same cluster.
2) NMSE of linear and non-linear ELM-AE and SELM-AE is lower than that of the TAE.
3) Linear and non-linear ELM-AE and SELM-AE learn features which are robust to noise.
4) Features learned by ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) are at least discriminative as TAE, PCA and NMF.
5) ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) features are sparser than PCA, but not sparse as NMF, hence this may contain holistic-based and parts-based features.
6) Training time of linear and non-linear ELM-AE and SELM-AE is lower than TAE.

II. Preliminaries

A. Extreme Learning Machines (ELM)

Feedforward neural networks are usually trained by Back-Propergation (BP) learning algorithm [34] in the past three decades. However BP faces bottlenecks such as slow learning speeds and local minimum problems. In contrast to the common understanding in neural networks community that all the hidden neurons in SLFNs need to be tuned, Extreme Learning Machine (ELM) theories [19]–[25] show that although the hidden nodes in SLFNs play critical roles they need not be tuned, instead they have universal approximation capability as long as these neurons are nonlinear piecewise continuous and randomly generated (e.g., random input weights and random biases in hidden nodes for additive neurons, or random centers and impact factors for Radial Basis Function neurons). This naturally overcomes the learning bottlenecks and barriers encountered by many conventional learning algorithms such as BP. Hence ELM has fast learning speed, and has both universal approximation and classification capabilities [19]–[24]. The network output of ELM network structure shown in Figure 1 is given by:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{1}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_L]^{\mathrm{T}}$ is the output weight matrix between the hidden nodes and the output nodes while $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})]$ is the hidden node output (random hidden feature) for the input $\mathbf{x} \in \mathbf{R}^d$, and $h_i(\mathbf{x})$ is the output of the i-th hidden node. For example, for additive hidden node $h_i(\mathbf{x}) = g(\mathbf{a}_i, \mathbf{x}, b_i)$, where $\mathbf{a}_i$ and b_i remain fixed after randomly generated. In ELM, the input data will be mapped to L dimensional ELM random feature space $\mathbf{h}(\mathbf{x})$. Given N training samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$, ELM is to resolve the following learning problems:

$$\text{Minimize: } ||\boldsymbol{\beta}||_p^{\sigma_1} + C||\mathbf{H}\boldsymbol{\beta} - \mathbf{T}||_q^{\sigma_2} \tag{2}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, $p, q = 0, \frac{1}{2}, 1, 2, \cdots, +\infty$, $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]^{\mathrm{T}}$ consists of the targets and $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \cdots, \mathbf{h}(\mathbf{x}_N)]^{\mathrm{T}}$. According to ELM learning theory, many types of feature mappings such as sigmoid, hard-limit, Gaussian, multi-quadric, Fourier series and wavelet, etc can be adopted in ELM. When C is infinitely large in Equation (2) the output weight $\boldsymbol{\beta}$ is calculated as:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse [37], [38] of matrix $\mathbf{H}$, which tends to achieve the smallest norm of $\boldsymbol{\beta}$ while keeping the training error to the minimum.

Minimum norm of output weights based ELM acheives a better generalization performance and a more robust solution [24], [25]. The output weights $\boldsymbol{\beta}$ from the ELM hidden layer to the output layer can be calculated as:

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{4}$$

or as:

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{5}$$

In detail, Huang et al. [24], [25] has shown that the equality constrained ELM given by Equation (4) and Equation (5) has fewer optimization constraints than Support Vector Machines (SVM) [39] and its variants including Least Square Support Vector Machine (LS-SVM) [40]. Unlike the SVM feature mapping function, ELM feature mapping $\mathbf{H}$ is non-parametric. In contrast to SVM, ELM provides a unified learning algorithm for regression, binary-class classification and multi-class classification with higher generalization performance and much faster learning speed [24], [25].

B. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [1], [2] aims to linearly project the $N \times d$ input data matrix $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T$ via an orthogonal transformation so that the first dimension in the projected space describes the most variance of the data while the second dimension of the projected data describes the second most variance, and so on. The first vector in the orthogonal transformation matrix is the first eigenvector which represents the variance described by the 1^{st} principal component. Hence, PCA can be used to reduce the dimensionality of data (from d to L where $d > L$) by removing the dimensions which describe the least variance of the data and preserving the dimensions which describe the most variance of the data. The 1^{st} eigenvector $\mathbf{V}_1$ is calculated as:

$$\mathbf{V}_1 = \underset{||\mathbf{V}_1||=1}{\text{Maximize}}\left(\frac{{\mathbf{V}_1}^T(\mathbf{X} - \text{mean}(\mathbf{X}))^T\,(\mathbf{X} - \text{mean}(\mathbf{X}))\,\mathbf{V}_1}{{\mathbf{V}_1}^T\mathbf{V}_1}\right) \tag{6}$$

According to Equation (6), $\mathbf{V}_1$ is maximized when $\mathbf{V}_1$ is the first eigenvector of data $\mathbf{X}_1$. Subsequent eigenvectors $\mathbf{V}_j$ are calculated as:

$$\hat{\mathbf{X}}_{j-1}^T = \mathbf{X} - \sum_{i=1}^{j-1}\mathbf{X}\mathbf{V}_{j-1}\mathbf{V}_{j-1}^T$$
$$\mathbf{V}_j = \underset{||\mathbf{V}||=1}{\text{Maximize}}\left(\frac{\mathbf{V}_j^T\hat{\mathbf{X}}_{j-1}^T\hat{\mathbf{X}}_{j-1}\mathbf{V}_j}{\mathbf{V}_j^T\mathbf{V}_j}\right) \tag{7}$$

Before performing PCA, the input data matrix $\mathbf{X}$ should be standardized to have zero mean. The projected lower dimensional representation of the input data $\mathbf{X}$ is created by multiplying the input data $\mathbf{X}$ with the selected eigenvectors $\mathbf{V}_j$ as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\mathbf{V}_j \tag{8}$$

C. Non-Negative Matrix Factorization (NMF)

Non-negative Matrix Factorization (NMF) [3] performs parts-based non-linear dimensionality reduction. NMF factorizes a positive data matrix $\mathbf{X}$ to a positive coefficient matrix $\mathbf{A}$ and a positive basis matrix $\mathbf{W}$. NMF factorization process is described as a square loss function as shown in Equation (9) or as a Kullback-Leibler divergence minimization function [41]. This paper adopts the following square loss which is commonly used in NMF:

$$\begin{aligned}&\text{Minimize:} \quad ||\mathbf{X} - \mathbf{A}\mathbf{W}||^2\\ &\text{Subject to:} \quad \mathbf{A} \geq 0,\ \mathbf{W} \geq 0\end{aligned} \tag{9}$$

The positive basis matrix $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_d]$ and positive coefficient matrix $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_L]$ are calculated using multiplicative update rules where k is the number of iterations:

$$\mathbf{A}^{k+1} = \mathbf{A}^k\frac{\mathbf{X}\mathbf{W}^T}{\mathbf{A}^k\mathbf{W}^{k+1}(\mathbf{W}^{k+1})^T}$$
$$\mathbf{W}^{k+1} = \mathbf{W}^k\frac{(\mathbf{A}^k)^T\mathbf{X}}{(\mathbf{A}^k)^T\mathbf{A}^k\mathbf{W}^k} \tag{10}$$

In contrast to PCA, matrices $\mathbf{W}$ and $\mathbf{A}$ in NMF are not orthogonal, hence the captured features are not uncorrelated. There are two main concerns when NMF is used: 1) NMF converges to a local minimum [41]; 2) using NMF for dimension reduction is mathematically unjustifiable as it has not been proved that NMF learns statistical information of input data.

The positive basis matrix $\mathbf{W}$ learns the most occurring parts in the data and the lower dimensional space $\mathbf{X}_{\text{proj}}$ is created by multiplying the input data $\mathbf{X}$ with the positive basis matrix $\mathbf{W}$ as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\mathbf{W}^T \tag{11}$$

D. Random Projection (RP)

Random Projection (RP) aims to reduce the dimension of data $\mathbf{X}$ while retaining the Euclidean information of data by projecting the data along an orthogonal [4] or sparse [5] random matrix $\mathbf{A}$:

$$||\mathbf{x}_i - \mathbf{x}_j|| \approx ||\mathbf{x}_i\mathbf{A} - \mathbf{x}_j\mathbf{A}|| \tag{12}$$

Orthogonal matrix $\mathbf{A}$ is represented as:

$$\mathbf{A}^T\mathbf{A} = \mathbf{I} \tag{13}$$

Sparse random matrix [5] is represented as:

$$a_{ij} = 1/\sqrt{L}\begin{cases} +\sqrt{3} & p = 1/6 \\ 0 & p = 2/3 \\ -\sqrt{3} & p = 1/6 \end{cases} \tag{14}$$

where p represents the ratio of elements in the sparse random matrix $\mathbf{A}$. Hence $\frac{2}{3}$ of the sparse random matrix values are zero, $\frac{1}{6}$ of the sparse random matrix values are $\sqrt{\frac{3}{L}}$, and $\frac{1}{6}$ of the sparse random matrix values are $-\sqrt{\frac{3}{L}}$. The time taken to generate sparse random weights are lower than orthogonal random weights [5] and the lower dimensional space $\mathbf{X}_{\text{proj}}$ is calculated as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\mathbf{A} \tag{15}$$

It is worth noting the difference between RP and ELM hidden layer feature mapping formed by random hidden neurons. Random projection uses linear projection which can only form a subspace of the original input space and does not have universal approximation capability. However, ELM uses non-linear hidden neurons which naturally has universal approximation and classification capabilities. The space formed by ELM feature mappings need not be subspace of the original input space, it can be a new space with non-linear relationship with the original input space. The new feature space formed by ELM feature mappings can have lower or higher dimensions than the original input space.

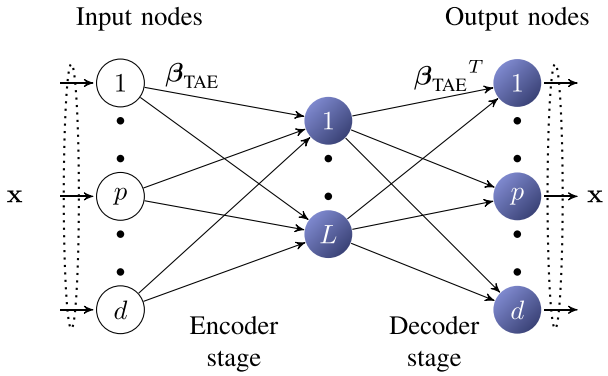
E. Tied Weight Auto-Encoder (TAE)

Similar to Restricted Boltzmann Machine (RBM) [42], Tied weight Auto-Encoder (TAE) [13] learns interesting features by using tied weights. TAE consists of an encoder stage and decoder stage as illustrated in Figure 2 and represents features of the input data in three different architectures: 1) compressed architecture; 2) sparse architecture; and 3) equal dimension architecture. (as shown in Figure 2).

Compressed architecture
: The number of input neurons is larger than the number of neurons in the hidden layer. In this case, TAE learns to capture features from the hidden layer feature space, which has a lower dimensionality than the input data space.

Sparse architecture
: The number of input neurons is smaller than the number of neurons in the hidden layer. In this case, TAE learns to capture features from the hidden layer feature space, which has a higher dimensionality than the input data space.

Equal dimension architecture
: The number of input neurons is equal to the number of neurons in the hidden layer. In this case, TAE learns to capture features from the hidden layer feature space, which has the same dimensionality as the input data space.



$d > L$: Compressed architecture
$d = L$: Equal dimension architecture
$d < L$: Sparse architecture

Fig. 2. TAE network architecture with tied weights $\boldsymbol{\beta}_{\text{TAE}}$.

As the main focus of this paper is dimension reduction compressed TAE architecture is discussed. The tied weight concept means: to use the same weights in the encoder stage and decoder stage of an Auto-Encoder (AE). This constrains the TAE to learn the same set of weights for dimension reduction and to reconstruct the data from the lower dimensional space.

The encoder stage of a TAE is given by:

$$\mathbf{Y} = g(\mathbf{X}\boldsymbol{\beta}_{\text{TAE}} + \mathbf{b}) \tag{16}$$

Sigmoid activation function is used in the encoder stage of the TAE and the decoder stage is given by:

$$\mathbf{Z} = g(\mathbf{Y}\boldsymbol{\beta}_{\text{TAE}}{}^T + \mathbf{c}) \tag{17}$$

where $\mathbf{c} = [c_1, \cdots, c_d]$ is the bias vector of the output nodes and $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_d]$ is the output of TAE. A linear activation function or sigmoid activation function is used in the decoder stage of TAE. TAE learning algorithm minimizes the squared error objective given by $L_2(\mathbf{X}, \mathbf{Z}) = ||\mathbf{X} - \mathbf{Z}||^2$. Dimension reduction is achieved in TAE by projecting the data $\mathbf{X}$ along the encoder stage weights $\boldsymbol{\beta}_{\text{TAE}}$ as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\boldsymbol{\beta}_{\text{TAE}} \tag{18}$$

III. ELM-Feature Representation With Random Hidden Neurons

Linear and non-linear Extreme Learning Machine Auto-Encoder (ELM-AE) and Sparse Extreme Learning Machine Auto-Encoder (SELM-AE) generates random hidden neurons (e.g, input weights and biases for additive neurons) in the encoder stage. Unlike TAE [42], the ELM-AE and SELM-AE do not use tied weights. As only the decoder

stage weights are calculated in ELM-AE and SELM-AE, the computational cost is lower than TAE. Linear and non-linear ELM-AE and SELM-AE encoder stage weights are randomly generated based on random projection theories [4], [5] to retain Euclidean distance information. However, whether a non-linear activation function can be used in non-linear ELM-AE and SELM-AE is supported by ELM theories [19]–[24]:

Lemma 1 Universal Approximation Capability [19]–[24]: Given any bounded non-constant piecewise continuous function as the activation function in hidden neurons, if by tuning parameters of hidden neuron activation function SLFNs can approximate any target continuous function, then for any continuous target function $f(\mathbf{x})$ and any randomly generated function sequence $\{h_i(\mathbf{x})\}_{i=1}^{L}$, $\lim_{L\to\infty} \| \sum_{i=1}^{L} \boldsymbol{\beta}_i h_i(\mathbf{x}) - f(\mathbf{x})\| = 0$ holds with probability one with appropriate output weights $\boldsymbol{\beta}$.

Lemma 1 shows that random hidden nodes (*nonlinear mapping* after *random projections*) can be used in different type of applications including clustering [43], [44], regression [23] and classification [24] by using different objective functions. For clustering the objective function is [44]:

$$\begin{aligned} &\underset{\boldsymbol{\beta}}{\text{Minimize}} \ ||\boldsymbol{\beta}||^2 + \lambda Tr(\boldsymbol{\beta}^T \mathbf{H}^T \mathbf{L} \mathbf{H} \boldsymbol{\beta}) \\ &\text{s.t.} \ (\mathbf{H}\boldsymbol{\beta})^T \mathbf{H}\boldsymbol{\beta} = \mathbf{I} \end{aligned} \tag{19}$$

where $\mathbf{L}$ is the graph Laplacian of input data $\mathbf{X}$ and λ is a user given parameter. For regression and classification the objective function is given by Equation (2).

Linear and non-linear ELM-AE and SELM-AE adopts an unsupervised learning as follows: input data are used as the output data: $\mathbf{t}$=$\mathbf{x}$, input weights and biases of the random additive hidden nodes are chosen to be orthogonal for ELM-AE; sparse input weights and biases are chosen for the random hidden neurons in SELM-AE.

A. Extreme Learning Machine Auto-Encoder (ELM-AE)

In contrast to the previous work of ELM-AE [36], this paper show that linear ELM-AE can learn the between-class scatter matrix, and also shows the efficacy of linear and non-linear ELM-AE in-terms of discriminative capability, sparsity, training time and Normalized Mean Square Error (NMSE). For compressed ELM-AE architecture, the hidden orthogonal random parameters project the input data to a lower dimension space and calculated by:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= g(\mathbf{x}\mathbf{A} + \mathbf{b}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})] \\ &= [g(\mathbf{a}_1 \cdot \mathbf{x} + b_1), \cdots, g(\mathbf{a}_L \cdot \mathbf{x} + b_L)] \\ \mathbf{A}^T\mathbf{A} &= \mathbf{I} \\ \mathbf{b}^T\mathbf{b} &= 1 \end{aligned} \tag{20}$$

The orthogonal random hidden parameters of linear and non-linear ELM-AE architecture are calculated by Equation (20) accordingly with $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_L]$ as the orthogonal random weight matrix between the input nodes and hidden nodes and $\mathbf{b} = [b_1, \cdots, b_L]$ is the bias vector of the hidden nodes. Vincent et al. [9] has shown that the hidden layer of an AE must retain information of the input data.

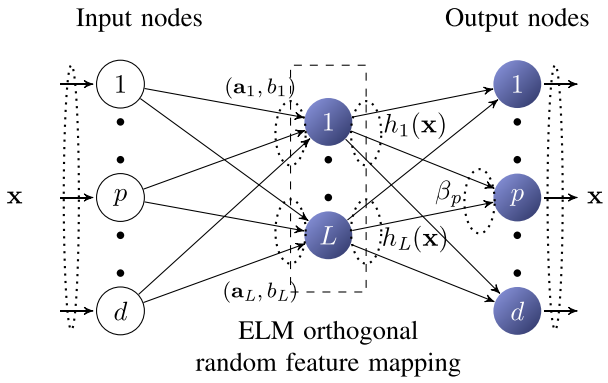


Fig. 3. Linear and non-linear ELM-AE and SELM-AE have the same solution as the original ELM except for: 1) The target output of linear and non-linear ELM-AE and SELM-AE is the same as input $\mathbf{x}$; 2) The hidden node parameters $g(\mathbf{a}_i, b_i)$ are made orthogonal after randomly generated for linear and non-linear ELM-AE, and the hidden node parameters $g(\mathbf{a}_i, b_i)$ are made sparse random for linear and non-linear SELM-AE. $h_i(\mathbf{x}) = g(\mathbf{a}_i, \mathbf{x}, b_i)$ is the output of i^{th} hidden node for input $\mathbf{x}$; 3) Linear activation function is used for linear ELM-AE and SELM-AE, while non-linear activation function is used for non-linear ELM-AE and SELM-AE.

Hence, orthogonal random parameters can be used in linear and non-linear ELM-AE to retain the Euclidean information of the input data as shown by Johnson-Lindenstrauss Lemma [4].

The output weights $\boldsymbol{\beta}_{\text{AE}}$ of linear and non-linear ELM-AE are responsible for the transformation from the feature space to input data ($\mathbf{t} = \mathbf{x}$) and satisfy:

$$\underset{\boldsymbol{\beta}_{\text{AE}}}{\text{Minimize:}} \ ||\mathbf{H}\boldsymbol{\beta}_{\text{AE}} - \mathbf{X}||^2 \tag{21}$$

The analytical solution of Equation (21) is given by Equation (3) when $\mathbf{T} = \mathbf{X}$.

The output weights $\boldsymbol{\beta}_{\text{AE}}$ of the compressed linear ELM-AE architecture representation with zero bias ($b_i = 0$) learn the between-class scatter matrix of data as shown in Theorem 2. The derivation is motivated by linear AE [6]. In contrast, linear AE and PCA learn the variance information of data while NMF learn a parts-based representation of data.

Theorem 2: Linear ELM-AE tries to minimize the objective function Minimize $_{\boldsymbol{\beta}_{AE}}$ $||\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{AE} - \mathbf{X}||^2$. When the number of input neurons in linear ELM-AE is larger than the number of hidden neurons ($d > L$), with zero bias ($b_i = 0$) and $\mathbf{A}^T\mathbf{A} = \mathbf{I}$; $\boldsymbol{\beta}_{AE} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T$, where $\mathbf{V}$ are the eigenvectors of covariance matrix $\mathbf{X}^T\mathbf{X}$.

Proof: As shown in Figure 3 in linear ELM-AE we try to have Minimize $_{\boldsymbol{\beta}_{\text{AE}}}$ $||\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} - \mathbf{X}||^2$, where $\boldsymbol{\beta}_{\text{AE}} = [\boldsymbol{\beta}_{\text{AE}1}, \cdots, \boldsymbol{\beta}_{\text{AE}L}]^T$ is the output weight matrix between the hidden nodes and the output nodes, $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_L]$ are the orthogonal random weight and $\{(\mathbf{x}_i)_{i=1}^{N}\}$ are the input and output data. It is reasonable to assume that the input data matrix $\mathbf{X}$ has full rank in most applications.

The normal equations of the convex function E is given by:

$$\mathbf{A}^T\mathbf{X}^T\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} = \mathbf{A}^T\mathbf{X}^T\mathbf{X}. \tag{22}$$

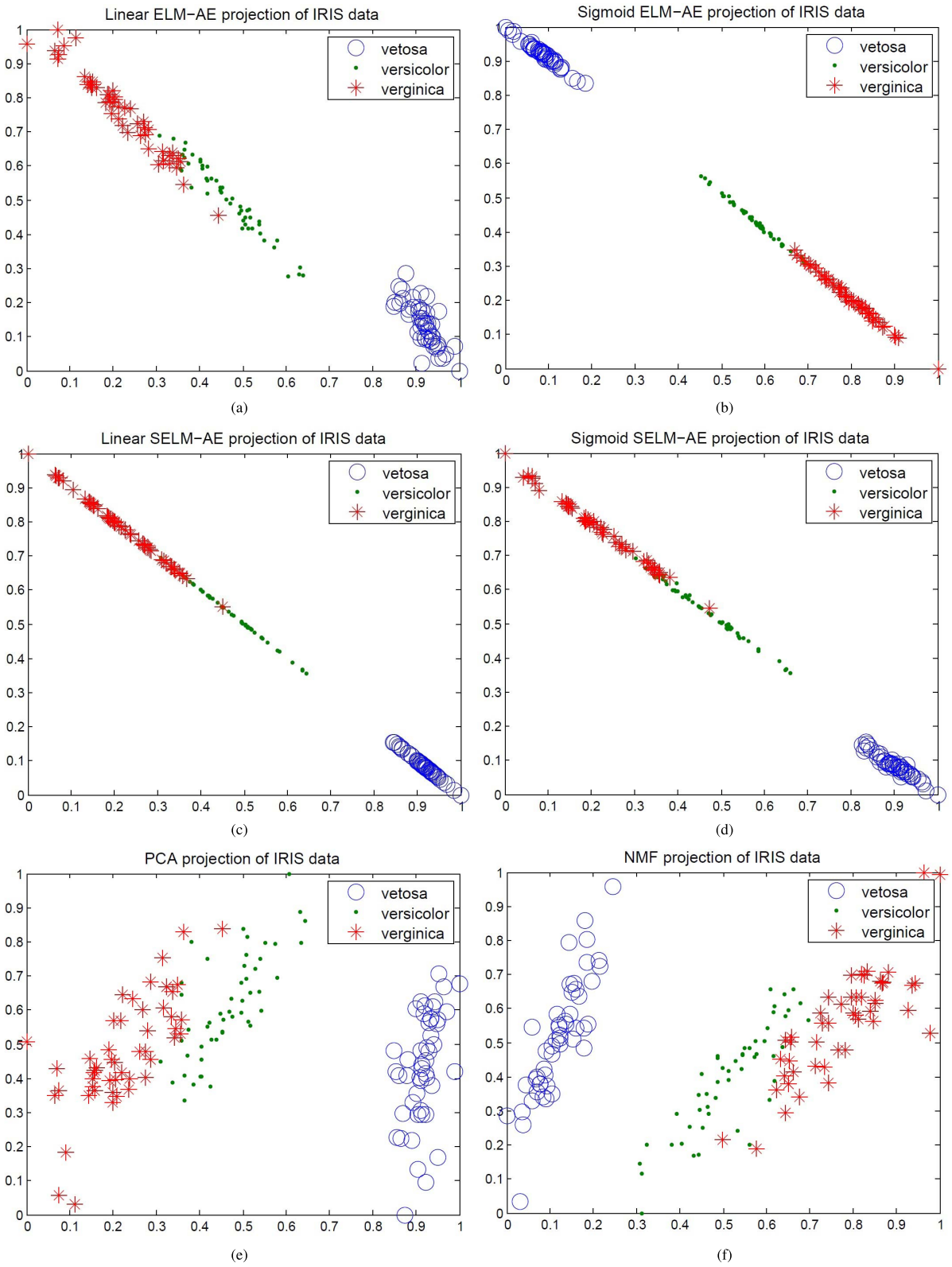


Fig. 4. The ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), PCA and NMF lower dimensional projections of IRIS dataset. (a) Output of projection $\mathbf{X_{IRIS}}\boldsymbol{\beta}^{\mathbf{T}}_{\mathbf{AE}}$ calculated by Equation (33). (b) Output of projection $\mathbf{X_{IRIS}}\boldsymbol{\beta}^{\mathbf{T}}_{\mathbf{AE}}$ calculated by Equation (33). (c) Output of projection $\mathbf{X_{IRIS}}\boldsymbol{\beta}^{\mathbf{T}}_{\mathbf{SAE}}$ calculated by Equation (37). (d) Output of projection $\mathbf{X_{IRIS}}\boldsymbol{\beta}^{\mathbf{T}}_{\mathbf{SAE}}$ calculated by Equation (37). (e) Output of projection $\mathbf{X_{IRIS}}\mathbf{V}^{\mathbf{T}}$ calculated by Equation (8). (f) Output of projection $\mathbf{X_{IRIS}}\mathbf{W}^{\mathbf{T}}$ calculated by Equation (11).

The $d > L$ orthogonal random matrix is created by performing Singular Value Decomposition (SVD) [45] on a $d \times d$ orthogonal random matrix where ${\mathbf{A}_d}^T\mathbf{A}_d = \mathbf{A}_d{\mathbf{A}_d}^T = \mathbf{I}$ and removing $d-L$ singular values. Hence based on Eckart-Young-Mirsky Theorem [46], [47] $\mathbf{A}\mathbf{A}^T$ is the best SVD low rank approximation of $\mathbf{A}_d{\mathbf{A}_d}^T$ and $\mathbf{A}\mathbf{A}^T \approx \mathbf{I}$. However, note that

$\mathbf{A}^T\mathbf{A} = \mathbf{I}$ for any d and L where $d > L$.

$$\mathbf{X}^T\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} = \mathbf{X}^T\mathbf{X} \tag{23a}$$

$$\mathbf{A} = \boldsymbol{\beta}_{\text{AE}}{}^T(\boldsymbol{\beta}_{\text{AE}}\boldsymbol{\beta}_{\text{AE}}{}^T)^{-1} \tag{23b}$$

From Equation (23b) and $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, we have:

$$\boldsymbol{\beta}_{AE} = \mathbf{A}^T\boldsymbol{\beta}_{\text{AE}}{}^T(\boldsymbol{\beta}_{\text{AE}}\boldsymbol{\beta}_{\text{AE}}{}^T)^{-1}\boldsymbol{\beta}_{\text{AE}} \tag{24}$$

In Equation (24) $P_{\boldsymbol{\beta}_{\text{AE}}} = \boldsymbol{\beta}_{\text{AE}}{}^T(\boldsymbol{\beta}_{\text{AE}}\boldsymbol{\beta}_{\text{AE}}{}^T)^{-1}\boldsymbol{\beta}_{\text{AE}}$ is the orthogonal projection matrix of $\boldsymbol{\beta}_{\text{AE}}$. Hence $\boldsymbol{\beta}_{\text{AE}} = \mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}}$ The covariance of input data $\mathbf{X}$ is $C_{\mathbf{XX}} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$, $\mathbf{V} = [\mathbf{V}_1 \cdots \mathbf{V}_d]$ and $\boldsymbol{\Sigma} = [\boldsymbol{\sigma}_1 \cdots \boldsymbol{\sigma}_d]$, where $\mathbf{V}_i$ and $\boldsymbol{\sigma}_i$ are the eigenvectors and eigenvalues of covariance matrix $\mathbf{X}^T\mathbf{X}$, respectively. The orthogonal projection $P_{\boldsymbol{\beta}_{\text{AE}}}$ have a relationship with the orthogonal projection $P_{\boldsymbol{\beta}_{\text{AE}}\mathbf{V}}$ as:

$$P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}} = \mathbf{V}^T\boldsymbol{\beta}_{\text{AE}}{}^T(\boldsymbol{\beta}_{\text{AE}}\mathbf{V}\mathbf{V}^T\boldsymbol{\beta}_{\text{AE}}{}^T)^{-1}\boldsymbol{\beta}_{\text{AE}}\mathbf{V} \tag{25}$$

By substituting $P_{\boldsymbol{\beta}_{\text{AE}}} = \boldsymbol{\beta}_{\text{AE}}{}^T(\boldsymbol{\beta}_{\text{AE}}\boldsymbol{\beta}_{\text{AE}}{}^T)^{-1}\boldsymbol{\beta}_{\text{AE}}$ and $\mathbf{V}\mathbf{V}^T = \mathbf{I}$ to Equation (25), we get the following relationship:

$$P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}} = \mathbf{V}^T P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{V} \tag{26a}$$

$$P_{\boldsymbol{\beta}_{\text{AE}}} = \mathbf{V}P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T \tag{26b}$$

Equation (23a) can be rearranged as:

$$\mathbf{X}^T\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} = \mathbf{X}^T\mathbf{X} \tag{27a}$$

$$C_{\mathbf{XX}}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} = C_{\mathbf{XX}} \tag{27b}$$

$$\boldsymbol{\beta}_{\text{AE}}{}^T\mathbf{A}^T C_{\mathbf{XX}}\mathbf{A}\boldsymbol{\beta}_{\text{AE}} = \boldsymbol{\beta}_{\text{AE}}{}^T\mathbf{A}^T C_{\mathbf{XX}} \tag{27c}$$

$\boldsymbol{\beta}_{\text{AE}}$ in Equation (27c) can be replaced with Equation (24) as:

$$P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{A}\mathbf{A}^T C_{\mathbf{XX}}\mathbf{A}\mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}} = P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{A}\mathbf{A}^T C_{\mathbf{XX}} \tag{28a}$$

$$(P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{A}\mathbf{A}^T C_{\mathbf{XX}}\mathbf{A}\mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}})^T = C_{\mathbf{XX}}\mathbf{A}\mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}} \tag{28b}$$

From Equation(28a) and Equation (28b), we get the following relationship:

$$\begin{aligned} P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{A}\mathbf{A}^T C_{\mathbf{XX}}\mathbf{A}\mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}} &= P_{\boldsymbol{\beta}_{\text{AE}}}\mathbf{A}\mathbf{A}^T C_{\mathbf{XX}} \\ &= C_{\mathbf{XX}}\mathbf{A}\mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}} \end{aligned} \tag{29}$$

By using Equation (29) and Equation (26b), we can get the following relationship:

$$\mathbf{V}P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T \tag{30a}$$

$$P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}\boldsymbol{\Sigma} = \boldsymbol{\Sigma}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}} \tag{30b}$$

In Equation (30b) $P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V}$ is diagonal matrix with rank L with eigenvalues 1 (L times) and eigenvalue 0 ($d - L$ times). Because $\boldsymbol{\Sigma}$ is a diagonal with eigenvalues $\boldsymbol{\sigma}_1 > \boldsymbol{\sigma}_2 > \cdots \boldsymbol{\sigma}_d > 0$. Hence, $P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T\mathbf{A}\mathbf{A}^T\mathbf{V} = \mathbf{I}_L$ with L number of one's (1) and $d - L$ zeros (0) in the diagonal.

$$P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}}\mathbf{V}^T\mathbf{A} = \mathbf{I}_L\mathbf{V}^T\mathbf{A} \tag{31}$$

Equation (31) shows that $P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}} = \mathbf{I}_L$. By substituting $P_{\boldsymbol{\beta}_{\text{AE}},\mathbf{V}} = \mathbf{I}_L$ and $\boldsymbol{\beta}_{\text{AE}} = \mathbf{A}^T P_{\boldsymbol{\beta}_{\text{AE}}}$ to Equation (26b), the output weights $\boldsymbol{\beta}_{\text{AE}}$ of linear ELM-AE is:

$$\boldsymbol{\beta}_{\text{AE}} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T \tag{32}$$

where $\mathbf{V}$ is a $d \times L$ matrix.

This completes the proof. □

Ding et al. [48] showed that between-class scatter matrix $\mathbf{M}\mathbf{M}^T$ can be represented by the eigenvector and its transpose $\mathbf{M}\mathbf{M}^T = \mathbf{V}\mathbf{V}^T$ (where $\mathbf{M} = (\mathbf{m}_1, \cdots, \mathbf{m}_m)$ and $\mathbf{m}_i$ is the mean of class i). Hence, learned features $\boldsymbol{\beta}_{\text{AE}} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T$ is the between-class scatter matrix $\mathbf{V}\mathbf{V}^T$ projected along the orthogonal random weights $\mathbf{A}$. Furthermore, Ding and He [48] has shown that the between-class scatter reduces the distance of the data points belonging to the same cluster. Hence, linear ELM-AE removes dimensions that least effect the Euclidean distance of data points and is essentially the lowest variance dimensions. However in contrast to PCA, linear ELM-AE groups data points belonging to the same cluster.

Dimension reduction is achieved in linear and non-linear ELM-AE by projecting the data $\mathbf{X}$ along the decoder stage weights $\boldsymbol{\beta}_{\text{AE}}$ as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\boldsymbol{\beta}_{\text{AE}}^{\text{T}} \tag{33}$$

B. Sparse Extreme Learning Machine Auto-Encoder (SELM-AE)

Sparsity is the main drive of creating a parts-based representation of an input data as shown in Sparse Coding [49], Reconstruction cost based Independent Component Analysis (RICA) [50] and NMF. Hence, for the proposed linear and non-linear SELM-AE, we impose sparsity by using sparse random parameters in the hidden layer. Furthermore, the hidden layer of an AE must retain information of the input data. To this end a sparse random matrix proposed by Achlioptas *et al*. [5] is used. Similar to orthogonal random matrix used in linear and non-linear ELM-AE, this sparse random matrix preserve the Euclidean distances between the data points in the projected space $\mathbf{XA}$ while introducing sparsity. Hence, in linear and non-linear SELM-AE the random hidden node parameters are calculated by:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= g(\mathbf{x}\mathbf{A} + \mathbf{b}) = [h_1(\mathbf{x}), \cdots, h_L(\mathbf{x})] \\ &= [g(\mathbf{a}_1 \cdot \mathbf{x} + b_1), \cdots, g(\mathbf{a}_L \cdot \mathbf{x} + b_L)] \\ a_{ij} = b_i &= 1/\sqrt{L} \begin{cases} +\sqrt{3} & p = 1/6 \\ 0 & p = 2/3 \\ -\sqrt{3} & p = 1/6 \end{cases} \end{aligned} \tag{34}$$

where $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_L]$ is the sparse random matrix between the input nodes and the hidden nodes and $\mathbf{b} = [b_1, \cdots, b_L]$ is the bias vector of the hidden nodes. The output weights $\boldsymbol{\beta}_{\text{SAE}}$ of linear and non-linear SELM-AE are calculated as:

$$\underset{\boldsymbol{\beta}_{\text{SAE}}}{\text{Minimize:}}\ ||\mathbf{H}\boldsymbol{\beta}_{\text{SAE}} - \mathbf{X}||^2 \tag{35}$$

Similar to ELM-AE, Theorem 3 shows that output weights $\boldsymbol{\beta}_{\text{SAE}}$ of linear SELM-AE learn the between-class scatter matrix.

Theorem 3: Linear SELM-AE tries to minimize the function $\underset{\boldsymbol{\beta}_{SAE}}{\text{Minimize:}}\ ||\mathbf{X}\mathbf{A}\boldsymbol{\beta}_{SAE} - \mathbf{X}||^2$. *When the number of hidden neurons in linear SELM-AE is smaller than the number of input neurons ($d > L$) and with zero bias ($b_i = 0$);* $\boldsymbol{\beta}_{SAE} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T$, *where* $\mathbf{V}$ *are the eigenvectors of covariance matrix* $\mathbf{X}^T\mathbf{X}$.

Proof: We first show that $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ as sparse random matrix $\mathbf{A}$ is not orthogonal.

By [5], we have $E[\mathbf{A}] = 0$ and $\mathbf{I} = var(\mathbf{A})$. Hence, $\mathbf{I} = var(\mathbf{A}) = E[\mathbf{A}^T\mathbf{A}] - (E(\mathbf{A}))^2$ and $\mathbf{I} = var(\mathbf{A}) = E[\mathbf{A}^T\mathbf{A}]$.

$$\mathbf{A}^T\mathbf{A} = cov(\mathbf{A}, \mathbf{A}) = E[\mathbf{A}^T\mathbf{A}] - E[\mathbf{A}^T]E[\mathbf{A}]$$
$$\mathbf{A}^T\mathbf{A} = cov(\mathbf{A}, \mathbf{A}) = \mathbf{I} \tag{36}$$

As $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and linear SELM-AE objective function is similar to linear ELM-AE objective function, we can to show that $\boldsymbol{\beta}_{\text{SAE}} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T$ by following the proof in Theorem 2. □

The learned features $\boldsymbol{\beta}_{\text{SAE}}$ (output weights) learn the between-class scatter matrix projected by sparse random weights $\boldsymbol{\beta}_{\text{SAE}} = \mathbf{A}^T\mathbf{V}\mathbf{V}^T$ which is sparser than features learned by linear ELM-AE.

Dimension reduction is achieved in linear and non-linear SELM-AE by projecting the data $\mathbf{X}$ along the decoder stage weights $\boldsymbol{\beta}_{\text{SAE}}$ as:

$$\mathbf{X}_{\text{proj}} = \mathbf{X}\boldsymbol{\beta}_{\text{SAE}}^{\text{T}} \tag{37}$$

C. Linear and non-Linear ELM-AE and SELM-AE Decoders Function as a Denoiser

Linear AE with hidden neurons equal to input neurons will learn the trivial identity function. To avoid learning trivial functions in AE the following constraints are applied: 1) applying a regularization term such as frobenius norm of the jacobian matrix in Contractive Auto-Encoders (CAE) [18] has shown to learn features which are invariant to small variations in the input data; 2) randomly corrupting the input data such as in Denoising Auto-Encoder (DAE) [9] and Marginalized Denoising Auto-Encoder (MDAE) [10] has shown to learn features robust to noise from the objectives $||g(\tilde{\mathbf{X}}\boldsymbol{\beta}^T)\boldsymbol{\beta} - \mathbf{X}||^2$ and $||\tilde{\mathbf{X}}\boldsymbol{\beta}^T - \mathbf{X}||^2$ respectively, where $\tilde{\mathbf{X}}$ is the randomly corrupted input data.

Linear and non-linear ELM-AE and SELM-AE also learn features robust to noise as the encoder stage randomly corrupts the input data $\mathbf{H} = \tilde{\mathbf{X}}$. To further elaborate, we describe the operations performed in the encoder stage ($\mathbf{H} = g(\mathbf{X}\mathbf{A} + \mathbf{b})$) of linear and non-linear ELM-AE and SELM-AE: 1) input data is multiplied with orthogonal (ELM-AE) or sparse random weights (SELM-AE) to retain the Euclidean distances which does not corrupt the data; 2) add random bias which corrupts the input data; 3) add non-linear activation function which corrupt the input data non-linearly (this step is performed only in non-linear ELM-AE and SELM-AE). Therefore, non-linear ELM-AE and SELM-AE decoders perform as a nonlinear denoiser while linear ELM-AE and SELM-AE decoders perform as a linear denoiser that learn features robust to noise by mapping the encoder stage output $\mathbf{H} = \tilde{\mathbf{X}}$ to input data $\mathbf{X}$. In contrast to DAE which adds noise in the input layer, linear and non-linear ELM-AE and SELM-AE add noise in the encoder. Furthermore, in contrast to MDAE which can only learn an equal dimension representation, linear and non-linear ELM-AE and SELM-AE can learn a compressed, equal dimension and sparse representation.

IV. Experiments

This paper shows that the proposed linear and non-linear ELM-AE and SELM-AE to an extend learns localized features and at least discriminative as TAE, PCA and NMF. Furthermore, the training time of linear and non-linear ELM-AE and SELM-AE is lower than TAE and NMF. Linear and non-linear ELM-AE and SELM-AE obtains lower NMSE between input and the reconstructed input data than TAE. To this end, this section compares the discriminative power, sparsity, training time and NMSE of the linear and non-linear ELM-AE, linear and non-linear SELM-AE, TAE, PCA and NMF features. The criteria for performance evaluations are as follows:

1) Discriminative capability measures the ability of the dimension reduction algorithm to learn important features and is measured by classifying the lower dimensional projections[2] using ELM classifier. As it has been shown that SVM is suboptimal to ELM classifier [24], [25]. Built in MATLAB functions for TAE, PCA and NMF are used.
2) Sparsity describes the dimension reduction algorithm capability to learn localized features[3] of the data and is measured by L_2/L_1 measure.
3) NMSE is calculated as: $\text{NMSE} = \sum_{i=1}^{N}(\mathbf{X}_i - \hat{\mathbf{X}}_i)^2/(N \times (\mathbf{X}_{\max} - \mathbf{X}_{\min}))$ where $\mathbf{X}_{\max}$ and $\mathbf{X}_{\min}$ is the largest value and smallest value of input data $\mathbf{X}$, respectively.

The experiments were carried out in a high-end computer with a Xeon E5-2650 2 GHz processor and 256 GB RAM running MATLAB 2015b.

A. Benchmark Datasets

TAE, PCA, NMF, ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) have been tested on USPS [51] handwritten digit recognition, CIFAR-10 object recognition [52] and NORB object recognition [53] datasets. These datasets were normalized to have zero mean and unit variance for TAE, PCA, ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear), while for NMF data was normalized between 0 and 1. USPS datatset contains 16×16 dimensional digits from 0 to 9. There are 7291 training examples and 2007 testing data samples.

CIFAR-10 dataset contains $3 \times 32 \times 32$ dimensional objects belonging to 10 classes. There are 40000 training data samples, 10000 samples for validation and 10000 testing data samples.

NORB data set contains $2 \times 96 \times 96$ dimensional object images of four legged animals, humans, airplanes, trucks and cars. There are 24300 training data samples and 24300 testing data samples. For our experiments we down sampled the NORB dataset to $2 \times 32 \times 32$.

B. Visual Investigation of Features Learned by ELM-AE and SELM-AE

Ding and He [48] showed that the between-class scatter matrix $\mathbf{M}\mathbf{M}^T$ could be represented by eigenvectors $\mathbf{M}\mathbf{M}^T = \mathbf{V}\mathbf{V}^T$, however, Ding and He [48] did not show whether it could be used for dimension reduction. This paper shows that between-class scatter with orthogonal random

[2]See Equation (8) Equation, (11), Equation (18), Equation (33) and Equation (37) to create lower dimension projections of PCA, NMF, TAE, ELM-AE (linear non-linear) and SELM-AE (linear and non-linear) respectively.

[3]Bengio et al. [7] showed the advantages of a parts-based representation.

TABLE I

THE PARAMETER RANGES USED TO DETERMINE THE BEST PARAMETERS OF ELM-AE (LINEAR AND NON-LINEAR), SELM-AE (LINEAR AND NON-LINEAR), TAE, PCA AND NMF

DataSet	The number of features	The number of hidden neurons for ELM	The L_2 norm parameter
USPS	$10, 20, \cdots, 240, 250$	$2000, 4000, 6000, 7000$	$1e-8, 1e-7, \cdots, 1e7, 1e8$
CIFAR-10	$100, 200, \cdots, 2900, 3000$	$10000, 20000, 30000, 40000$	$1e-8, 1e-7, \cdots, 1e7, 1e8$
NORB	$100, 200, \cdots, 1900, 2000$	$10000, 20000, 24000$	$1e-8, 1e-7, \cdots, 1e7, 1e8$

TABLE II

THE TESTING ACCURACY, TRAINING TIME AND NMSE OF USPS, CIFAR-10 AND NORB DATASET OF ELM-AE (LINEAR AND NON-LINEAR), SELM-AE (LINEAR AND NON-LINEAR), TAE, PCA AND NMF

DataSet	Algorithm	Features	Testing Accuracy (%)	Training Time (s)	NMSE
USPS	ELM-AE (linear)	40	98.71(± 0.1)	0.084	0.03(±0)
	ELM-AE (sigmoid)	40	98.7(± 0.11)	0.065	0.03(±0)
	SELM-AE (linear)	50	98.71 (± 0.12)	0.058	0.06(±0)
	SELM-AE (sigmoid)	40	98.77(± 0.12)	0.078	0.06(±0)
	TAE [13]	180	98.33 (± 0.21)	31.71	0.20(±0.01)
	PCA [1], [2]	20	98.46 (± 0.13)	0.054	0.027(±0)
	NMF [3]	60	98.56 (± 0.16)	40.35	0.06(±0)
CIFAR-10	ELM-AE (linear)	300	53.43(± 0.26)	6.3	0.01(±0)
	ELM-AE (sigmoid)	300	53.65(± 0.28)	6.5	0.06(±0)
	SELM-AE (linear)	300	53.48 (± 0.32)	6.2	0.01(±0)
	SELM-AE (sigmoid)	300	53.92 (± 0.3)	6.3	0.01(±0)
	TAE [13]	400	46.78 (± 0.18)	3484.1	0.23(±0.001)
	PCA [1], [2]	100	53.32 (± 0.36)	9.1	0.07(±0)
	NMF [3]	100	51.75 (± 0.34)	3361	0.063(±0.048)
NORB	ELM-AE (linear)	1000	91.58(± 0.3)	12.78	0.0001(±0.02)
	ELM-AE (sigmoid)	1000	91.76(± 0.26)	12.29	0.0003(±0)
	SELM-AE (linear)	1000	91.54 (± 0.37)	11.65	0.0001(±0)
	SELM-AE (sigmoid)	1000	91.69 (± 0.26)	11.65	0.0006(±0)
	TAE [13]	700	85.76 (± 0.33)	4149.5	0.26(±0.003)
	PCA [1], [2]	100	88.25 (± 0.28)	2.63	0.003(±0)
	NMF [3]	100	86.54 (± 0.69)	84.69	0.33(±0.002)

neurons or sparse random neurons can be used for dimension reduction. Iris dataset [54] can be used to visually show that lower dimensional projections of ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) cluster the data belonging to the same cluster together. The iris dataset consists of four dimensions and three classes representing the species of iris flower named sentosa, versicolor and verginica. The spread of the data in each cluster of ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) projections (as shown in Figure 4(a), Figure 4(b), Figure 4(c) and Figure 4(d)) is smaller than the spread of PCA and NMF projection (as shown in Figure 4(e) and Figure 4(f)), which verify that the ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) projection clusters data belonging to the same cluster appropriately.

C. Measuring Sparsity of the Learned Features

This paper uses L_2/L_1 [55] sparsity measure to evaluate the sparsity of the learned features. Hence, L_2/L_1 measure indicates sparsity at an abstract level. L_2/L_1 measure is a ratio which weighs the features, so that large coefficients in the features get a small weight and small coefficients get a large weight. A higher L_2/L_1 measure indicates that there are few large feature coefficient values and more small feature coefficient values. A high L_2/L_1 value represents that the algorithm learns local information of data (parts-based features), a low value represents that the algorithm learns global information of data (holistic-based features), and a moderate value represents that the algorithm learn both local and global information of data.

D. Measuring Discriminative Power of the Learned Features

The discriminative power of the learned features of TAE, PCA, NMF, ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) were tested by reducing the dimensions of USPS, CIFAR-10 and NORB datasets (training and testing) using the following steps:

$k =$ Number of features to be extracted

1) Calculate the $\boldsymbol{\beta}_{\text{AE}}$, $\boldsymbol{\beta}_{\text{SAE}}$, $\boldsymbol{\beta}_{\text{TAE}}$, $\mathbf{V}$ and $\mathbf{W}$ using ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE,, PCA and NMF for k with training data.
2) Create lower dimensional training data and testing data for ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE, PCA and NMF projections by Equation (33), Equation (37), Equation (18), Equation (8) and Equation (11), respectively.

The average classification accuracy of thirty trials are reported for each ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE, PCA and NMF algorithm.

E. Performance Comparison of Different Algorithms in USPS, CIFAR-10 and NORB Datasets

Table II shows the average testing accuracy for features learned by ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE, PCA and NMF. The number of parameters to be selected are the number of features, the number

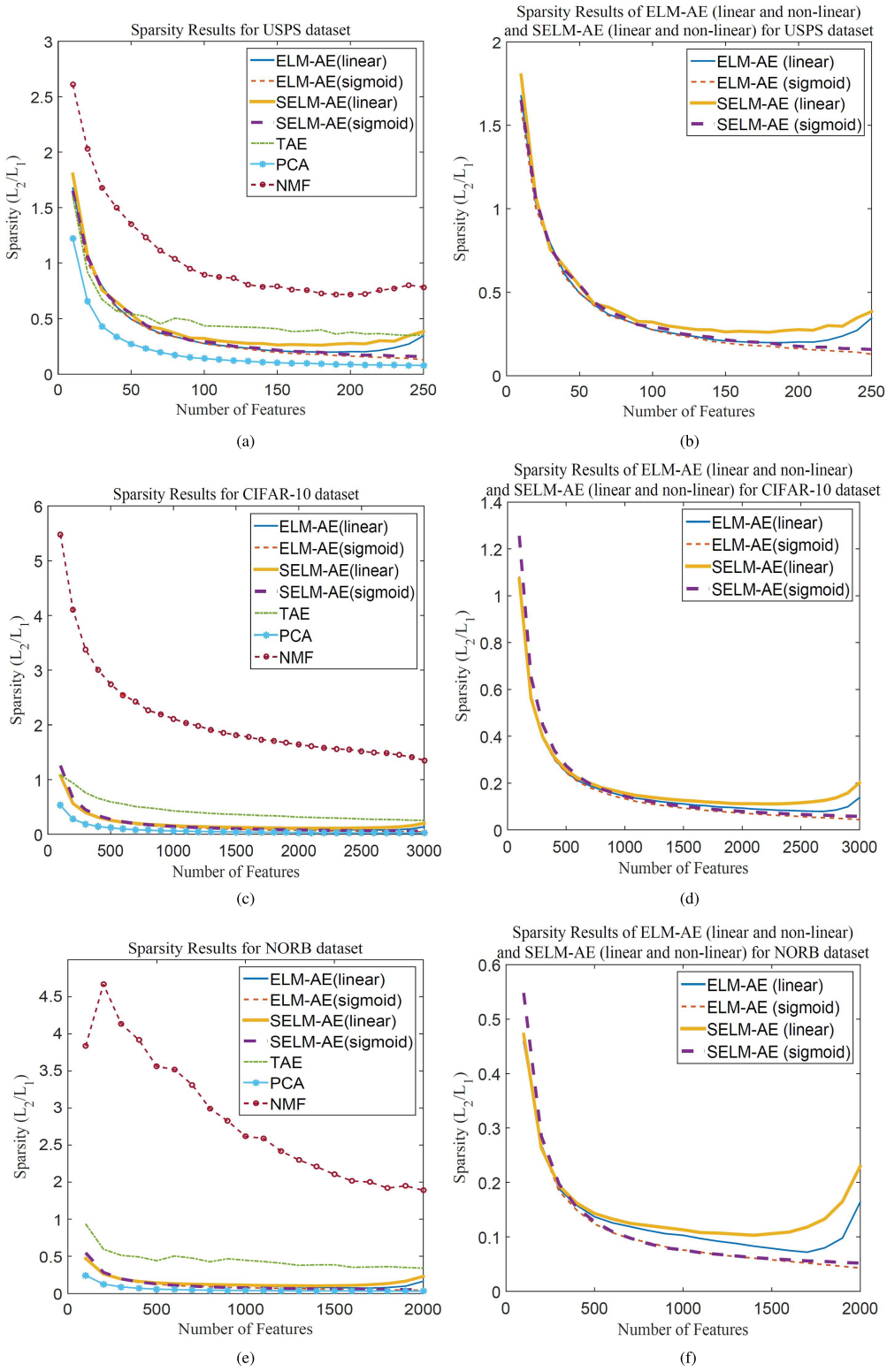


Fig. 5. Sparsity values of ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE, PCA and NMF for USPS, CIFAR-10 and NORB datasets. (a) ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) is sparser than PCA for USPS dataset. (b) Large features of linear SELM-AE is sparser than linear ELM-AE, non-linear SELM-AE and non-linear ELM-AE for USPS dataset. (c) ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) is sparser than PCA for CIFAR-10 dataset. (d) Large features of linear SELM-AE is sparser than linear ELM-AE, non-linear SELM-AE and non-linear ELM-AE for CIFAR-10 dataset. (e) ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) is sparser than PCA for NORB dataset. (f) Large features of linear SELM-AE is sparser than linear ELM-AE, non-linear SELM-AE and non-linear ELM-AE for NORB dataset.

of hidden neurons of the ELM classifier and the L_2 norm parameter of the ELM classifier for USPS, CIFAR-10 and NORB dataset. Table I shows the range of the parameters used to find the best accuracy of ELM-AE (linear and non-linear), SELM-AE (linear and non-linear), TAE, PCA and NMF.

Table II shows that non-linear SELM-AE lower dimensional projection allows ELM to classify 9 (0.44%) more samples than TAE projection, 8 (0.3%) more samples than PCA projection and 4 (0.2%) samples than NMF projection for USPS dataset. However, TAE projection require nearly 4 times more features than non-linear SELM-AE projection. For CIFAR-10 dataset non-linear SELM-AE lower dimensional projection allows ELM to classify 714 (7.14%) more samples than TAE projection, 60 (0.6%) more samples than PCA projection and 217 (2.17%) more samples than NMF projection. For NORB dataset non-linear ELM-AE lower dimensional projection allows ELM to classify 1458 (6%) more samples than TAE projection, 842 (3.51%) more samples than PCA projection and 1253 (5.22%) more samples than NMF projection.

The training time of linear and non-linear ELM-AE and SELM-AE low dimensional projection is at least 200 times faster than TAE and at least 8 times faster than NMF for USPS, CIFAR-10 and NORB datasets. However the training time of linear and non-linear ELM-AE and SELM-AE is in general equal to PCA in USPS and CIFAR-10 datasets but 5 times slower in NORB dataset.

NMSE values shows that linear and non-linear ELM-AE and SELM-AE is at least 4 times lower than TAE and similar to PCA for USPS, CIFAR-10 and NORB datasets. Hence, non-linear ELM-AE and SELM-AE learn a better model than TAE.

The corresponding sparsity values of ELM-AE, SELM-AE, TAE, PCA and NMF features for USPS, CIFAR-10 and NORB dataset is shown in Figure 5. Figure 5 shows that ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) are sparser than PCA for USPS, CIFAR-10 and NORB datasets, but not as sparse as NMF. Hence, linear and non-linear ELM-AE and SELM-AE learn local (parts-based) and global (holistic-based) information of data. Furthermore, Figure 4(b), Figure 4(d) and Figure 4(f) show that linear SELM-AE is sparser than linear ELM-AE, non-linear SELM-AE and non-linear ELM-AE for large features.

PCA algorithm learns global information (holistic-based) of the data, while NMF learns local information (parts-based) of data. In contrast TAE, linear and non-linear ELM-AE and SELM-AE learn both local and global information as they try to maximize the mutual information [9]. Hence they require more features than PCA and NMF.

V. Conclusions

Conventional AE tunes input weights and output weights iteratively to learn features of data. This paper investigates linear and non-linear ELM-AE and SELM-AE with orthogonal and sparse random hidden neurons. Orthogonal random and sparse random input neurons (e.g., random input weights and biases in additive neurons) are used to retain the Euclidean information of the data in the hidden layer [4], [5] as an AE must retain information of the data in the hidden layer [9]. As only the output weights must be calculated computational complexity of the proposed linear and non-linear ELM-AE and SELM-AE is lower.

In contrast to the common perception that linear AE learn the variance information of the data, the proposed linear ELM-AE and linear SELM-AE with random neurons in theory learn the between-class scatter matrix which reduces the distances of data points belonging to the same cluster.

NMSE values shows that non-linear ELM-AE and SELM-AE learn a better model than TAE.

Non-linear ELM-AE and SELM-AE hidden layer output is calculated in three steps: 1) use orthogonal or sparse random weights to project the data to a lower dimension projection which retains Euclidean information of the data; 2) add orthogonal or sparse random bias to the lower dimension projection which corrupts the Euclidean information of the data; 3) add a non-linear activation function to the output of the previous step which corrupts the output of previous step non-linearly. Hence, the decoding stage of non-linear ELM-AE and SELM-AE tries to learn features which are robust to noise by trying to map the output of the hidden layer to the data.

The experimental results show that the proposed ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) learn features more localized than PCA (higher sparsity than PCA), while not localized as NMF (lower sparsity than NMF). Furthermore, ELM-AE (non-linear and linear) and SELM-AE (linear and non-linear) features are at least discriminative as TAE, PCA and NMF, while PCA features are more discriminative than NMF. Hence this paper shows that ELM-AE (linear and non-linear) and SELM-AE (linear and non-linear) learn features which are discriminative and sparse.

References

[1] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 6, pp. 559–572, 1901.

[2] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Edu. Psychol.*, vol. 24, no. 6, pp. 417–441, 1933.

[3] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[4] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," *Contemp. Math.*, vol. 26, pp. 189–206, 1984.

[5] D. Achlioptas, "Database-friendly random projections," in *Proc. 20th Symp. Principles Database Syst.*, 2001, pp. 274–281.

[6] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.

[7] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[9] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[10] M. Chen, Z. Xu, K. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proc. 29th Int. Conf. Mach. Learn.*, Jul. 2012, pp. 767–774.

[11] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Neural Inf. Process. Syst.*, 1993, pp. 3–10.

[12] Y. Le Cun, "Modèles connexionnistes de l'apprentissage [Connectionist Models of Learning]," Ph.D. Thesis, University of Paris, 1987.

[13] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19. 2007, pp. 153–160.

[14] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proc. Neural Inf. Process. Syst.*, 2006, pp. 1137–1144.
[15] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20. 2008, pp. 873–880.
[16] I. J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, and A. Y. Ng, "Measuring invariances in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 22. 2009, pp. 646–654.
[17] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 536–543.
[18] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 833–840.
[19] 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, Jul. 2006.
[20] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, 2007.
[21] R. Zhang, Y. Lan, G.-B. Huang, and Z.-B. Xu, "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, Feb. 2012.
[22] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3460–3468, Oct. 2008.
[23] 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.
[24] 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.
[25] G.-B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Cognit. Comput.*, vol. 6, no. 3, pp. 376–390, 2014.
[26] G.-B. Huang, "What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cognit. Comput.*, vol. 7, no. 3, pp. 263–278, 2015.
[27] 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.
[28] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, May 2015.
[29] D. L. Sosulski, M. L. Bloom, T. Cutforth, R. Axel, and S. R. Datta, "Distinct representations of olfactory information in different cortical centres," *Nature*, vol. 472, pp. 213–216, Apr. 2011.
[30] C. Eliasmith *et al.*, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.
[31] O. Barak, M. Rigotti, and S. Fusi, "The sparseness of mixed selectivity neurons controls the generalization–discrimination trade-off," *J. Neurosci.*, vol. 33, no. 9, pp. 3844–3856, 2013.
[32] M. Rigotti *et al.*, "The importance of mixed selectivity in complex cognitive tasks," *Nature*, vol. 497, pp. 585–590, May 2013.
[33] S. Fusi, E. K. Miller, and M. Rigotti, "Why neurons mix: High dimensionality for higher cognition," *Current Opinion Neurobiol.*, vol. 37, pp. 66–74, Apr. 2016.
[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[35] 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, Mar. 1998.
[36] 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, Dec. 2013.
[37] D. Serre, *Matrices: Theory and Applications*. New York, NY, USA: Springer-Verlag, 2002.
[38] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York, NY, USA: Wiley, 1971.
[39] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[40] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.
[41] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 556–562.
[42] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[43] G. Huang, T. Liu, Y. Yang, Z. Lin, S. Song, and C. Wu, "Discriminative clustering via extreme learning machine," *Neural Netw.*, vol. 70, pp. 1–8, Oct. 2015.
[44] G. Huang, S. Song, J. N. D. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2405–2417, Dec. 2014.
[45] G. H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix," *J. Soc. Ind. Appl. Math. B, Numer. Anal.*, vol. 2, no. 2, pp. 205–224, 1965.
[46] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936.
[47] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *Quart. J. Math.*, vol. 11, no. 1, pp. 50–59, 1960.
[48] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 29–37.
[49] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, 1997.
[50] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng, "CA with reconstruction cost for efficient overcomplete feature learning," in *Proc. Neural Inf. Process. Syst.*, 2011, pp. 1017–1025.
[51] A. Demiriz, K. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 225–254, 2002.
[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[53] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jun. 2004, pp. II-97–II-104.
[54] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.
[55] N. Hurley and S. Rickard, "Comparing measures of sparsity," *IEEE Trans. Inf. Theory*, vol. 55, no. 10, pp. 4723–4741, Oct. 2009.

Liyanaarachchi Lekamalage Chamara Kasun received the B.Sc. degree from the Sri Lanka Institute of Information Technology, Sri Lanka, and the M.Sc. degree from Nanyang Technological University, Singapore, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include extreme learning machines, multilayer neural networks, and dimension reduction.

Yan Yang received the Ph.D. degree from the Faculty of Engineering and the Environment, University of Southampton. She is currently a Senior Research Fellow with the Energy Research Institute, Nanyang Technological University. Her research interests include machine learning, human–machine interaction, and system engineering.

Guang-Bin Huang (M'98–SM'04) received the B.Sc. degree in applied mathematics and the M.Eng. degree in computer engineering from Northeastern University, China, in 1991 and 1994, respectively, and Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 1999. During undergraduate period, he also concurrently studied with the Applied Mathematics Department and Wireless Communication Department, Northeastern University, China. He serves as an Associate Editor of *Neurocomputing*, *Neural Networks*, *Cognitive Computation*, and the IEEE TRANSACTIONS ON CYBERNETICS. His current research interests include machine learning, computational intelligence, and extreme learning machines.

He was a Research Fellow with the Singapore Institute of Manufacturing Technology, from 1998 to 2001, where he has led/implemented several key industrial projects (e.g., a Chief Designer and a Technical Leader of Singapore Changi Airport Cargo Terminal Upgrading Project). From 2001, he has been an Assistant Professor and an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He received the best paper award of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS in 2013.

Zhengyou Zhang (SM'97–F'05) received the Ph.D. and D.Sc. degrees in computer science from the University of Paris XI. He is currently a Principal Researcher and a Research Manager of the Multimedia, Interaction, and Communication Group with Microsoft Corporation. His research interests include computer vision, speech signal processing, multisensory fusion, multimedia computing, real-time collaboration, and human–machine interaction. He is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT.