Scientia Iranica D (2020) 27(6), 3005–3018

SCIENTIA IRANICA

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
http://scientiairanica.sharif.edu

ML-CK-ELM: An efficient multi-layer extreme learning machine using combined kernels for multi-label classification

M. Rezaei Ravari[a], M. Eftekhari[a,*], and F. Saberi-Movahed[b]

a. *Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.*
b. *Department of Applied Mathematics, Faculty of Sciences and Modern Technologies, Graduate University of Advanced Technology, Kerman, Iran.*

Received 5 May 2019; received in revised form 1 March 2020; accepted 15 June 2020

KEYWORDS
Extreme learning machine;
Kernel learning;
Multi-label learning;
Neural network;
Linear combination.

Abstract. Recently, many neural network methods have been proposed for multi-label classification in the literature. One of these recent methods is the Multi-Layer Extreme Learning Machines (ML-ELMs) in which stack auto encoders are used for tuning their weights. However, ML-ELMs suffer from three primary drawbacks: First, input weights and biases are chosen randomly; second, the pseudoinverse solution for calculating output weights will increase the reconstruction error; third, memory and execution time of transformation matrices are proportional to the number of hidden layers. In this paper, Multi-Layer Kernel Extreme Learning Machine (ML-CK-ELM) that uses a linear combination of base kernels in each layer is proposed for multi-label classification. The proposed approach effectively addresses the above-mentioned drawbacks. Furthermore, multi-label classification data are inherently characterized by multi-modal aspects due to a variety of labels assigned to each instance. Applying a combination of different kernels is the added advantage of ML-CK-ELM that implicitly assesses the inherent multi-modal aspects of multi-label data; each kernel can be effectively used to cover one of the modals better than other kernels. The empirical study indicates that ML-CK-ELM shows competitively better performance than other state-of-the-art methods, and experimental results of multi-label datasets verify the feasibility of ML-CK-ELM.

1. Introduction

Classification is an important technology for performing data analysis, which is vastly used in machine learning, data mining, pattern recognition, etc. In conventional classification, each sample corresponds to a unique label. This model is called "single-label classification". Currently, many investigations have been conducted on single-label classification such as support vector machine, decision tree, naive Bayes method, and neural network. Nevertheless, with the advent of more efficient technology, a single-label classification algorithm cannot deal with a large amount of labeled data. Compared to the single-label classification problem, each sample in the multi-label classification task is associated with more than one label. For example, a natural scene [1] can simultaneously correspond to mountain, tree, and sunset; a movie may correspond to several genres. Multi-label classification is widely used in image annotation [2] text classification and many other areas [1–4]. Let

*. *Corresponding author.*
E-mail addresses: mohammad.rezaei@eng.uk.ac.ir (M. Rezaei Ravari); m.eftekhari@uk.ac.ir (M. Eftekhari); f.saberimovahed@kgut.ac.ir (F. Saberi-Movahed)

doi: 10.24200/sci.2020.53490.3263

$X = \mathbb{R}^d$ denote a d-dimensional input space of features and each instance from X is assigned to a subset of labels $\lambda \subseteq Y$, where $Y = \{y_1, y_2, \cdots, y_q\}$ is a finite set of labels with $|Y| = q$. Hence, multi-label classification is the task of mapping inputs x to binary vectors y. Existing methods of multi-label classification are categorized into two groups [3]: the problem transformation methods and the problem adaption methods. In transformation methods, the multi-label classification problem converts into some single-label classification tasks such as binary relevance [4], random k-label sets [5], and classifier chain [6]. Generally, the "transformation" methods ignore the correlation between labels. On the other hand, the algorithm adaptation methods adopt learning techniques to deal with multi-label data directly. Representative algorithms include Multi-Label k-Nearest Neighbor (ML-KNN), which was proposed by Zhang and Zhou [7], and it employs the maximum posterior probability of the k-nearest neighbor samples to label prediction. Zhang [8] extended the Radial Basis Function (RBF) to deal with multi-label problems by performing the k-means clustering algorithm to determine the centers of the RBF functions and the number of hidden layer nodes in the network. The Regularized Extreme Learning Machine (RELM) [9] was introduced to improve the robustness of Extreme Learning Machine (ELM). Zhang et al. [10] proposed a deep network called multi-label extreme learning machine with ML-ELM-RBF for multi-label classification. Moreover, ML-ELM-RBF staked ELMs based on Auto Encoder (ELM-AE) in the first layers is followed by a final layer of RBF for classification. Lately, a ML-ELM [11–13] was proposed in which multiple layers of ELM-AE were used for representation learning in the first layers and then, classification was employed in the last layer, as shown in Figure 1.

Furthermore, the weight of the output layer can be determined in the closed form, while the input parameters were randomly generated [14]. A significant advantage of the ML-ELM is its fast training rate and efficient generalization. From the literature [15], using kernels in ELM shows very good efficiency without the need for random parameter assignment. Accordingly, a kernel version of ML-ELM called Multi-Layer Kernel ELM (ML-KELM) [16] was proposed for representation learning in which the outputs of hidden layers were encoded in the form of kernel matrix without the need for tuning the input parameters. The key points of the ML-KELM method can be listed as follows:

1. Randomly generated parameters (i.e., input weights and bias) are not required. To be more specific, the ML-KELM method makes use of an efficient projection framework in order to produce an optimal solution;
2. The ML-KELM method uses a non-singular transformation matrix and avoids the reconstruction error being generated by pseudoinverse solution;
3. ML-KELM utilizes only two transformation matrices and therefore, memory and execution time of transformation matrices can be reduced significantly.

The multi-label classification due to multi-modal aspects of multi-label data requires a suitable kernel to make a classifier more stable. However, the use of a specific kernel may be a source of bias and therefore, allowing a classifier to consider a set of kernels may lead to finding a better solution. In such a case, combining kernels is one possible way to make use of multiple information sources.

In this paper, considering the idea of linear combinations of kernels, a novel Multi-Layer Extreme Learning Machine using Combined Kernels (ML-CK-ELM) is adopted to solve multi-label classification problems so that a linear combination of some base kernels can be used in each layer. For this purpose, a specific kernel is constructed in which some predefined kernels (such as linear, sigmoid, and RBF kernels) are combined to improve the performance of the multi-label classification. In addition, a multi-label kernelized ELM model is proposed. In this regard, given that our proposed model takes advantage of the ML-KELM model, it exhibits a good level of robustness. Experimental results on several different multi-label datasets justify that the proposed algorithm generally outperforms other state-of-the-art methods.

The remainder of this paper is structured as follows. Section 2 briefly reviews the ML-ELM and introduces the ML-KELM. The proposed method is described in Section 3. In Section 4, experimental results demonstrate the capabilities of ML-KELM. Eventually, the conclusions are drawn in the last section.

2. Related work

This section briefly introduces the fundamental concepts and requirements that are used in our proposed approach.

2.1. Multi-layer Extreme Learning Machine (ML-ELM)

Suppose that $X^{(i)} = [x_1^{(i)}, \cdots, x_n^{(i)}]^T$ indicates the input matrix, where $x_k^{(i)}$ is the ith data representation for the input x_k, $i = 1, \cdots, L$, and $k = 1, \cdots, n$. Let $H^{(i)}$ be the output matrix of the ith hidden layer node with respect to $X^{(i)}$. In ML-ELM, the ith transformation matrix $\Gamma^{(i)}$ is considered as $\Gamma^{(i)} = [\gamma_1^{(i)}, \cdots, \gamma_n^{(i)}]$, where $\gamma_k^{(i)}$ denotes the transformation vector used for representation learning with respect to $x_k^{(i)}$. In addition, the matrix $\Gamma^{(i)}$ satisfies the following

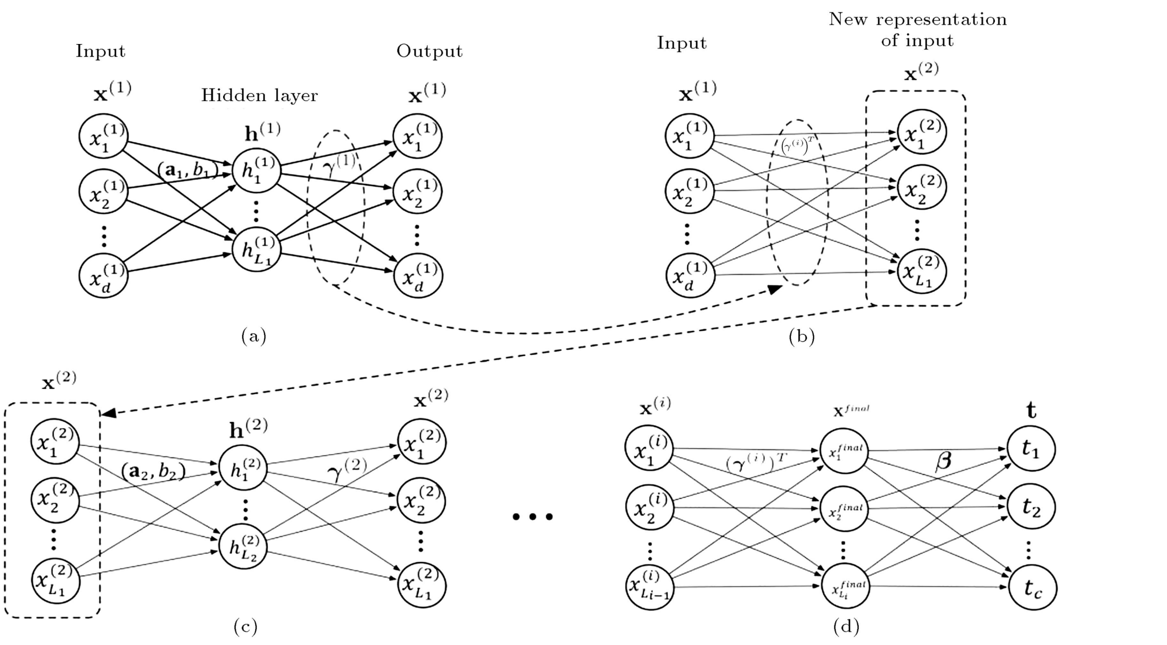

Figure 1. The architecture of Multi-Layer Extreme Learning Machine (ML-ELM) [16]: (a) The transformation matrix $\gamma^{(1)}$ of ELM-AE for transformation learning, (b) the new representation input $\mathbf{x}^{(2)}$ obtained by $g(x^{(1)}(\gamma^{(1)})^T)$, (c) $\mathbf{x}^{(2)}$ the input of ELM-AE for the next data representation, and (d) $\mathbf{x}^{(final)}$ used to compute the output weight β for classification upon finishing the representation learning.

equation:

$$H^{(i)}\Gamma^{(i)} = X^{(i)}, \tag{1}$$

where $H^{(i)}$ is defined by:

$$H^{(i)} = \begin{bmatrix} g_1^{(i)}\left(a_1^{(i)}, b_1^{(i)}, x_1^{(i)}\right) & \cdots & g_L^{(i)}\left(a_L^{(i)}, b_L^{(i)}, x_1^{(i)}\right) \\ \vdots & \ddots & \vdots \\ g_1^{(i)}\left(a_1^{(i)}, b_1^{(i)}, x_n^{(i)}\right) & \cdots & g_L^{(i)}\left(a_L^{(i)}, b_L^{(i)}, x_n^{(i)}\right) \end{bmatrix},$$

in which $g_i^{(i)}(a^{(i)}, b^{(i)}, x^{(i)})$ is an optional activation function used in the ith layer, and both the input weight $a^{(i)}$ and the bias $b^{(i)}$ are initialized randomly. Accordingly, Γ^i can be obtained as follows:

$$\Gamma^{(i)} = (H^{(i)})^{\dagger} X^{(i)}, \tag{2}$$

where $(H^{(i)})^{\dagger}$ is the Moore-Penrose pseudo-inverse of $H^{(i)}$. In this regard, in order to enhance the robustness of ELM, the Frobenius norm of $\Gamma^{(i)}$ can be considered as a regularization term to constrain $\Gamma^{(i)}$. Therefore, the objective function of ELM can be represented as follows:

$$\min_{\Gamma^{(i)}} \frac{C}{2}\left\|X^{(i)} - H^{(i)}\Gamma^{(i)}\right\|_F^2 + \frac{1}{2}\left\|\Gamma^{(i)}\right\|_F^2, \tag{3}$$

where C is the regularization constant and specified by users. Here, it can be shown that the solution $\Gamma^{(i)}$ can be determined by:

$$\Gamma^{(i)} = \begin{cases} \left(\frac{I}{C} + (H^{(i)})^T(H^{(i)})\right)^{-1}(H^{(i)})^T X^{(i)}, & n \geq L \\ (H^{(i)})^T\left(\frac{I}{C} + (H^{(i)})(H^{(i)})^T\right)^{-1} X^{(i)}, & n < L \end{cases} \tag{4}$$

where I represents an identity matrix.

The aim of using the transformation matrix $\Gamma^{(i)}$ is representation learning and, as shown in Figure 1, a new data representation $X^{(i+1)}$ can be calculated by multiplying $X^{(i)}$ by $\Gamma^{(i)}$ as follows:

$$X^{(i+1)} = g\left(X^{(i)}\left(\Gamma^{(i)}\right)^T\right). \tag{5}$$

Then, the final data representation of the first data representation $X^{(1)}$ is obtained. To be more specific, X^{final} is used as a hidden layer output to calculate the output β. Indeed:

$$\min_{\beta} \frac{C}{2}\left\|T - X^{final}\beta\right\|_F^2 + \frac{1}{2}\|\beta\|_F^2, \tag{6}$$

in which the output β is calculated by:

$$\beta = \begin{cases} \left(\frac{I}{C} + \left(X^{final}\right)^T X^{final}\right)^{-1}\left(X^{final}\right)^T T, & n \geq L \\ \left(X^{final}\right)^T\left(\frac{I}{C} + X^{final}\left(X^{final}\right)^T\right)^{-1} T, & n < L \end{cases} \tag{7}$$

The general framework of ML-ELM is summarized in Figure 1.

2.2. Multi-Layer Kernel Extreme Learning Machine (ML-KELM)

Wong et al. [16] introduced ML-KELM for representation learning, which is the integrated form of kernel learning and ML-ELM. The framework of ML-KELM consists of two separate learning procedures. The first procedure is the unsupervised representation learning by stacking the Kernel ELM based on Auto Encoder (KELM-AE). The second procedure is classification task (supervised procedure) using KELM. Similar to ELM-AE, KELM-AE learns the data transformation from the hidden layer to output. To this end, as shown in Figure 2, by applying a kernel function to the input matrix $X^{(i)}$, the kernel matrix $\Omega^{(i)}$ is obtained. Then, similar to the relation given in Eq. (1), the ith transformation matrix $\Gamma^{(i)}$ in KELM-AE is learned as follows:

$$\Omega^{(i)}\Gamma^{(i)} = X^{(i)}, \tag{8}$$

where $\Gamma^{(i)}$ can be obtained via the exact inverse of the kernel matrix rather than the pseudoinverse as follows:

$$\Gamma^{(i)} = \left(\frac{I}{C} + \Omega^{(i)}\right)^{-1} X^{(i)}. \tag{9}$$

Here, it should be noted that the matrix $\Omega^{(i)}$ is a square matrix, and adding the term $\frac{I}{C}$ guarantees that the matrix $\frac{I}{C} + \Omega^{(i)}$ be invertible (in fact, symmetric positive definite). For this reason, the exact inverse of $\frac{I}{C} + \Omega^{(i)}$ exists, and the kernel matrix methods avoid the reconstruction error being generated by the pseudoinverse solution. Therefore, the reconstruction error is reduced and the data representation $X^{(i+1)}$ is calculated as follows:

$$X^{(i+1)} = g\left(X^{(i)}\left(\Gamma^{(i)}\right)^T\right). \tag{10}$$

In the supervised procedure, the final data representation X^{final}, which is obtained from the previous procedure, is applied as the input in order to train K-ELM classification, i.e.:

$$\Omega^{final}\beta = T, \tag{11}$$

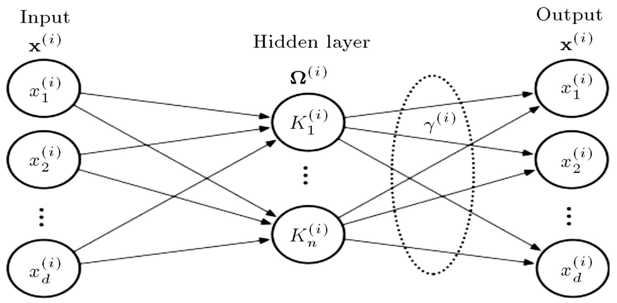

Figure 2. The architecture of the ith Kernel Extreme Learning Machine-based Auto Encoder (KELM-AE) [16], in which the hidden layer is replaced by a kernel matrix $\mathbf{\Omega}^{(i)}$.

where Ω^{final} is the kernel matrix, which is computed based on X^{final}, and the output weight β can be computed as follows:

$$\beta = \left(\frac{I}{C} + \Omega^{final}\right)^{-1} T. \tag{12}$$

Let us assume that $Z^{(1)} = [z_1^{(1)}, \cdots, z_m^{(1)}]^T \in \mathbb{R}^{m\times d}$ denotes a set of m test samples. In the first step, the data representation $Z^{(i+1)}$ is obtained by multiplying the ith transformation matrix $\Gamma^{(i)}$:

$$Z^{(i+1)} = g\left(Z^{(i)}\left(\Gamma^{(i)}\right)^T\right). \tag{13}$$

Then, the final data representation $Z^{final} = [z_1^{final}, \cdots, z_m^{final}]^T$ is obtained and applied to calculating the test kernel matrix $\Omega_z \in \mathbb{R}^{m\times n}$ whose entries $(\Omega_z)_{k,j}$ are determined using the following relation:

$$(\Omega_z)_{k,j} = K\left(z_k^{final}, x_j^{final}, \sigma^{final}\right), \tag{14}$$

where x_j^{final} is the jth data point from X^{final} and the kernel function K can be defined as follows:

$$K\left(z_k^{final}, x_j^{final}, \sigma^{final}\right) = \exp\left(-\frac{\left\| z_k^{final} - x_j^{final}\right\|_2}{2\left(\sigma^{final}\right)^2}\right). \tag{15}$$

In the end, the output of the network $\tilde{Y}$ is given by:

$$\tilde{Y} = \Omega_z\beta. \tag{16}$$

According to [12], if the ith and $(i+1)$th layers have the same dimension, the activation function g can be chosen as a linear piecewise and given that each $\Gamma^{(i)}$, for $i = 2, \cdots, N$, is a square matrix of the dimension n, a single unified transformation matrix $\Gamma_{unified} = \Gamma^{(i)}.\Gamma^{(i-1)}\cdots\Gamma^{(2)}$ can be obtained in ML-KELM so that only the two transformation matrices $\Gamma^{(1)}$ and $\Gamma_{unified}$ are required. From a practical point of view, the final data representation is directly determined as $Z^{(final)} = g(Z^{(1)}(\Gamma^{(1)})^T).\Gamma^T_{unified}$. Here, it is worthwhile to mention that this relation helps resolve the problem of memory storage and improve the execution time.

2.3. Motivation for linear kernel combination

Multi-label classification data inherently has multi-modal aspects due to the various labels assigned to each instance [17–20]. A combination of different kernels can be used to implicitly assess the inherent multi-modal aspects of multi-label data. In fact, each kernel can be effectively applied to cover modals that are better than the other kernels. Due to the recently performed researches regarding the multi-modal aspects of multi-label classification tasks and owing to the capabilities of different kernels for performing classification, a fusion of different kernels is utilized in this paper. To be more specific, different linear combinations of three kernels, namely linear, sigmoid, and RBF, are considered in different layers of the ML-ELM network. With this end of view, let $\{K_1, \cdots, K_p\}$ be a finite set of kernels combined to define more complex kernels. There are different ways that the combination can be performed. In this paper, the linear combination is used such that the combination function can be linearly parameterized as follows:

$$K(x_i, x_j) = \sum_{m=1}^{P} w_m K_m(x_i^m, x_j^m), \tag{17}$$

where each weight w_m can be considered as a real number, and it is assumed that the sum of all w_m's is one, i.e.:

$$\sum_{m=1}^{P} w_m = 1.$$

Then, it is replaced in the KELM method to make it suitable for multi-label classification. For the sake of simplicity, the values of w_m for different layers are coming from the set $\{0, 0.1, 0.2, 0.3, \cdots, 1\}$. This assumption helps tune the parameters w_m through the cross validation process.

3. Proposed method

The flowchart of our proposed method is given in Figure 3. As it can be seen, the first box indicates the inputs of our algorithm including training and test data as well as the initial values of some parameters like the regularization parameter C. To begin with, ML-KELM procedure is called for the second box to perform the training process of the ML-CK-ELM method. Then, the training processes $X^{(final)}$, $\Gamma^{(1)}$, $\Gamma_{unified}$, and β are calculated. Afterward, in the test stage, the data representation $Z^{(final)}$ associated with the test data is obtained by using Eq. (12) in order to calculate the test kernel matrix Ω_z. The model output is calculated by Eq. (15) and finally, a zero constant threshold is employed to predict the labels of each sample. Indeed, the label is set to 1, if the output label $\tilde{Y}$ is greater than zero. Moreover, it is set to -1 otherwise.

In ML-KELM, the transformation matrix $\Gamma^{(i)}$ and the input matrix $X^{(i)}$ are obtained using KELM-AE and then, the final kernel matrix Ω^{final} is calculated with respect to X^{final} and is used as the input for training. Finally, the output weight β is obtained using Eq. (11). Figure 4 details the procedure of ML-KELM.

In KELM-AE, the input matrix $X^{(i)}$ is mapped into a kernel matrix Ω^i by a weighted linear combination of base kernels sets. Then, the ith transformation

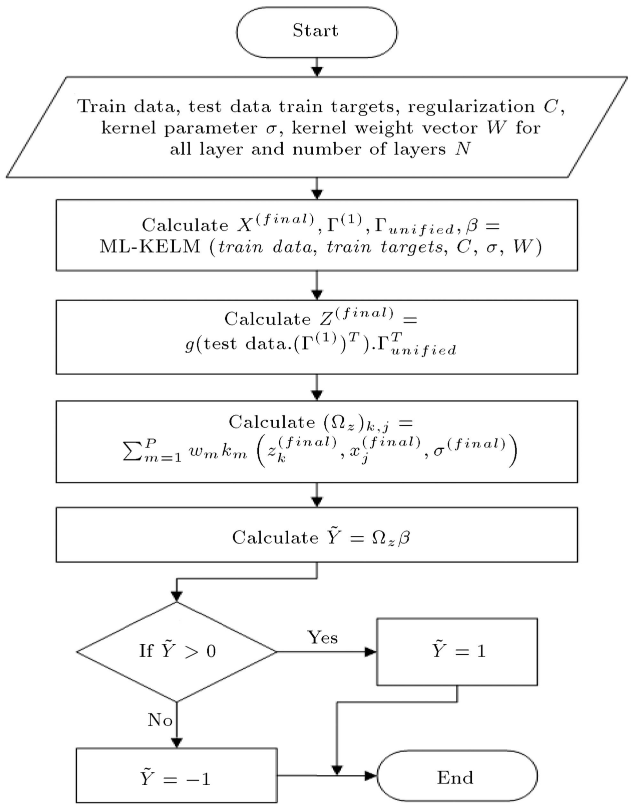
Start
Train data, test data train targets, regularization C, kernel parameter σ, kernel weight vector W for all layer and number of layers N
Calculate $X^{(final)}, \Gamma^{(1)}, \Gamma_{unified}, \beta =$ ML-KELM (*train data, train targets*, C, σ, W)
Calculate $Z^{(final)} = g(\text{test data}.(\Gamma^{(1)})^T).\Gamma^T_{unified}$
Calculate $(\Omega_z)_{k,j} = \sum_{m=1}^{P} w_m k_m \left(z_k^{(final)}, x_j^{(final)}, \sigma^{(final)}\right)$
Calculate $\tilde{Y} = \Omega_z \beta$
If $\tilde{Y} > 0$
Yes
$\tilde{Y} = 1$
No
$\tilde{Y} = -1$
End

Figure 3. The flowchart of Multi-Layer Combined Kernel Extreme Learning Machine (ML-CK-ELM).

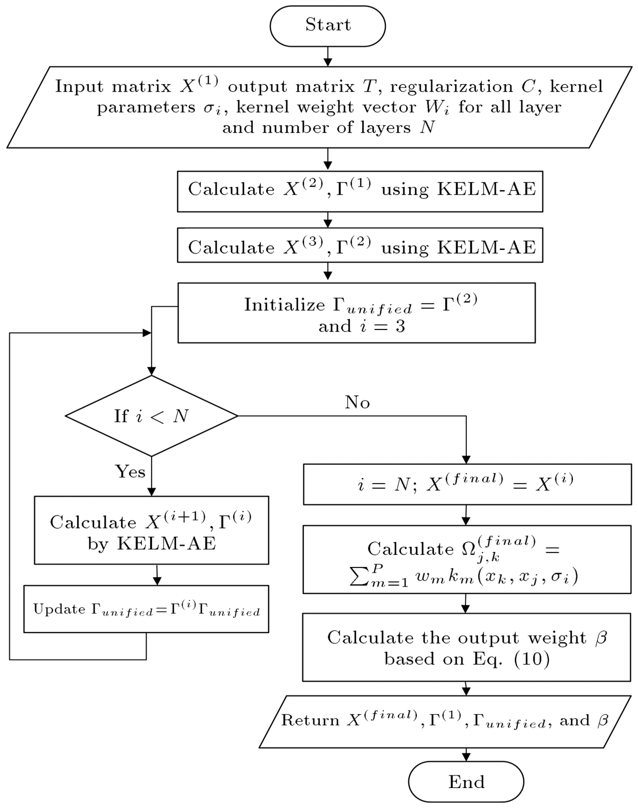
Start
Input matrix $X^{(1)}$ output matrix T, regularization C, kernel parameters σ_i, kernel weight vector W_i for all layer and number of layers N
Calculate $X^{(2)}, \Gamma^{(1)}$ using KELM-AE
Calculate $X^{(3)}, \Gamma^{(2)}$ using KELM-AE
Initialize $\Gamma_{unified} = \Gamma^{(2)}$ and $i = 3$
If $i < N$
No
Yes
Calculate $X^{(i+1)}, \Gamma^{(i)}$ by KELM-AE
Update $\Gamma_{unified} = \Gamma^{(i)}\Gamma_{unified}$
$i = N$; $X^{(final)} = X^{(i)}$
Calculate $\Omega^{(final)}_{j,k} = \sum_{m=1}^{P} w_m k_m(x_k, x_j, \sigma_i)$
Calculate the output weight β based on Eq. (10)
Return $X^{(final)}, \Gamma^{(1)}, \Gamma_{unified}$, and β
End

Figure 4. The training procedure via Multi-Layer Kernel Extreme Learning Machine (ML-KELM) algorithm.

matrix $\Gamma^{(i)}$ can be obtained by Eq. (8). In the last step of KELM-AE, the new data representation X^{i+1} is calculated by Eq. (9). The implementation of KELM-AE is depicted in Figure 5.

4. Experimental results and discussion

In this section, extensive experiments were conducted to evaluate the performance of the proposed method, which is compared with other relevant state-of-the-art methods.

4.1. Performance metrics for comparison

Experiments in this paper employed the following five evaluation indicators [4], which are extensively used in multi-label learning:

1. **Hamming loss:** Evaluating how many times an example-label pair is misclassified:

$$Hamming\ loss\ (h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{q} \left| h(x_i) \Delta y_i \right|, \quad (18)$$

   where Δ stands for the symmetric difference between the two sets;

2. **One-error:** Evaluating how many times the top-ranked label is not in the set of relevant labels:

$$One\ error\ (h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{q} \left[\left[\arg\max_{y \in Y} (f(x_i, y) \right] \notin Y_i \right]. (19)$$

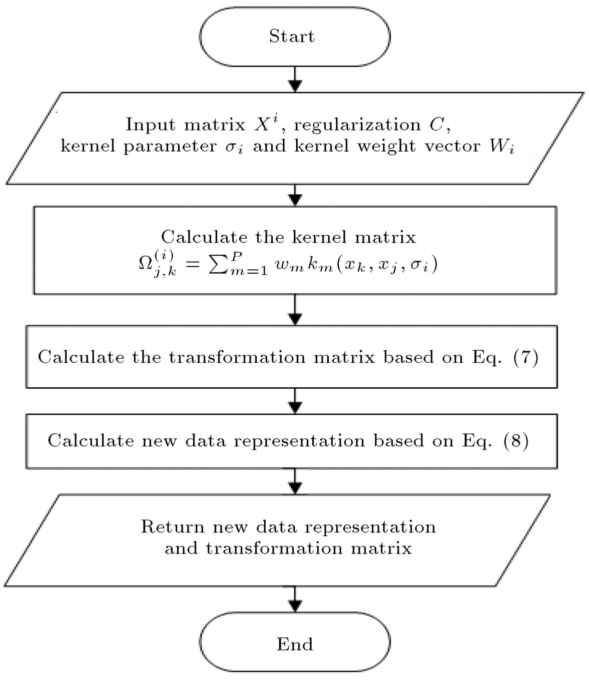
Start
Input matrix X^i, regularization C, kernel parameter σ_i and kernel weight vector W_i
Calculate the kernel matrix $\Omega^{(i)}_{j,k} = \sum_{m=1}^{P} w_m k_m(x_k, x_j, \sigma_i)$
Calculate the transformation matrix based on Eq. (7)
Calculate new data representation based on Eq. (8)
Return new data representation and transformation matrix
End

Figure 5. The algorithm of Kernel Extreme Learning Machine-based Auto Encoder (KELM-AE) for the ith layer.

   For any predicate $\pi, [[\pi]]$ if π satisfied returns 1, otherwise 0;

3. **Coverage:** Evaluating how many steps, on average, are needed to go down the ranked list of labels to cover all relevant labels;

$$Covrage\ (h) = \frac{1}{m'} \sum_{i=1}^{m'} \max_{y \in Y_i} \text{rank}_f(x_i, y) - 1. \quad (20)$$

4. **Ranking loss:** Evaluating the average fraction of label pairs that are reversely ordered.

$$Rankin\ gloss\ (h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{|Y_i| \, |\overline{Y_i}|} \left| \{ (y', y'') | \, f(x_i, y') \le f(x_i, y''), (y', y'') \in Y_i \times \overline{Y_i} \} \right|, \quad (21)$$

   where $\overline{Y_i}$ denotes the complementary set of Y_i in the label space Y;

5. **Average precision:** Evaluating the average fraction of relevant labels ranked above a particular label:

$$Average\ precision\ (h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{|Y_i|} \sum_{y \in Y_i} \frac{|y'| \, \text{rank}_f (x, y') \le \text{rank}_f(x, y), y' \in Y_i|}{\text{rank}_f(x_i, y)}. \quad (22)$$

Note that for the first four metrics, the smaller the values, the better the performance, while for the last one, the larger the values, the better the performance. These metrics serve as a good indicator for comprehensive comparative studies as they evaluate the performance of the learned models from various aspects.

4.2. Comparison methods

To investigate the efficiency of the proposed ML-CK-ELM method, we compare its classification performance with the following state-of-the-art and high-performance methods:

- **ML-KNN [7] (2007):** Multi-label k-nearest neighbors proposed by Zhang et al., employing the maximum posterior probability of the k-nearest neighbor samples to label prediction;
- **ML-RBF [8] (2009):** Multi-Label Radial Basis Function proposed by Zhang et al., extending the RBF to deal with multi-label problems by performing the k-means clustering algorithm with the aim of determining the centers of the RBF functions and the number of hidden layer nodes in the network;

- **RELM** [9] **(2010):** Regular Extreme Learning Machine proposed by Deng et al., employing the structural risk minimization and weighted least square to improve the robustness of ELM;
- **ML-ELM-RBF** [10] **(2016):** Multi-Label Extreme Learning Machine with Radial Basis Function proposed by Zhang et al., adopting the idea of radial basis function for multi-label learning (ML-RBF) and weight uncertainty ELM-AE.

4.3. Experiments setup

The experiments are conducted using Python 3.6 running on a 2.4-GHz i7 CPU with 8 GB RAM. In this paper, 5-fold Cross Validation (5-CV) is used for parameter tuning and the regularization parameter C is set as 2^r, $\{r = -10, -9, \cdots, 10\}$ in each layer. RBF, linear, sigmoid, and polynomial kernels are combined to form a fused kernel function. The kernel parameter σ for RBF is set as 2^r, $\{r = -6, -5, \cdots, 4\}$ in each layer. Moreover, the weight of kernel w_m is chosen from the set $\{0, 0.1, 0.2, 0.3, \cdots, 1\}$. Moreover, based on cross-validation results, in most of datasets, the average weight of RBF kernel is greater than the others in all layers. Moreover, for some datasets, the weights of RBF and Polynomial kernels are equal.

The experiments were conducted on fourteen publicly available benchmark datasets. The detailed information about all the datasets is summarized in Table 1.

4.4. Evaluations and performance analysis

The results in Tables 2 to 15 illustrating the average of different metrics over test data for ML-ELM-RBF, ML-RBF, ML-KNN, and RELM were adopted from [10]. The experimental results include the average of different performance measures for 10 runs over test data. In all tables, the indicator ↓ in front of one measure means that the smaller value for that measure is the better performance. In contrast, the ↑ sign denotes that the higher value for one measure is superior to the smaller values. From the results shown in Table 2,

Table 1. Properties of benchmark datasets.

Data set	**Number of training samples**	**Number of test samples**	**Features**	**Labels**
Yeast	1500	917	103	14
Scene	2000	407	294	6
Delicious	12920	3185	500	983
Art	2000	3000	462	26
Business	2000	3000	438	30
Computer	2000	3000	681	33
Education	2000	3000	550	33
Entertainment	2000	3000	640	21
Health	2000	3000	612	32
Recreation	2000	3000	606	22
Reference	2000	3000	793	33
Science	2000	3000	743	40
Social	2000	3000	1047	39
Society	2000	3000	636	27

Table 2. Average test results of five multi-label algorithms on yeast dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	**0.1896**	0.1899	0.1978	0.1962	0.1985
One-error↓	0.2356	**0.2297**	0.2372	0.2334	0.2356
Coverage↓	**6.1985**	6.2674	6.4963	6.4089	6.5691
Ranking-loss↓	**0.1599**	0.1632	0.1736	0.1726	0.1776
Average-precision↑	**0.7702**	0.7673	0.7586	0.7588	0.7555

Table 3. Average test results of five multi-label algorithms on scene dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	ML-ELM-RBF	ML-RBF	ML-KNN	RELM
Hamming-loss↓	**0.1095**	0.1653	0.1614	0.1884	0.1664
One-error↓	**0.2867**	0.2920	0.2916	0.3480	0.2908
Coverage↓	**0.5402**	0.8751	0.8865	1.0320	0.8929
Ranking-loss↓	**0.0973**	0.1521	0.1553	0.1923	0.1564
Average-precision↑	**0.8299**	0.8121	0.8112	0.7738	0.8105

Table 4. Average test results of five multi-label algorithms on delicious dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	ML-ELM-RBF	ML-RBF	ML-KNN	RELM
Hamming-loss↓	**0.0166**	0.0177	0.0178	0.0184	0.0178
One-error↓	0.3437	**0.3252**	0.3344	0.4085	0.3358
Coverage↓	**547.0185**	607.723	685.995	551.428	615.649
Ranking-loss↓	**0.1175**	0.1216	0.1410	0.1186	0.1235
Average-precision↑	0.3773	**0.3820**	0.3779	0.3276	0.3756

Table 5. Average test results of five multi-label algorithms on art dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	ML-ELM-RBF	ML-RBF	ML-KNN	RELM
Hamming-loss↓	0.0567	**0.0537**	0.0542	0.0585	0.0545
One-error↓	**0.4703**	0.4730	0.4759	0.5467	0.4803
Coverage↓	**4.5873**	5.4506	5.7788	4.7650	5.5812
Ranking-loss↓	**0.1157**	0.1401	0.1498	0.1262	0.1434
Average-precision↑	**0.6258**	0.6122	0.6066	0.5717	0.6071

Table 6. Average test results of five multi-label algorithms on business dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	ML-ELM-RBF	ML-RBF	ML-KNN	RELM
Hamming-loss↓	**0.0253**	0.0254	0.0255	0.0267	0.0254
One-error↓	0.1163	**0.1150**	0.1146	0.1210	0.1168
Coverage↓	2.4087	2.5083	2.7733	**2.1407**	2.5430
Ranking-loss↓	0.0396	0.0418	0.0463	**0.0359**	0.0422
Average-precision↑	**0.8863**	0.8844	0.8805	0.8822	0.8823

Table 7. Average test results of five multi-label algorithms on computer dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	ML-ELM-RBF	ML-RBF	ML-KNN	RELM
Hamming-loss↓	0.0347	**0.0345**	**0.0345**	0.0373	0.0351
One-error↓	**0.3420**	0.3549	0.3585	0.4083	0.3610
Coverage↓	4.0757	4.7178	5.0634	**3.9317**	4.6083
Ranking-loss↓	0.0823	0.0976	0.1058	**0.0790**	0.0949
Average-precision↑	**0.7109**	0.6988	0.6960	0.6670	0.6995

Table 8. Average test results of five multi-label algorithms on education dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	**0.0375**	0.0377	**0.0375**	0.0397	0.0377
One-error↓	**0.4597**	0.4735	0.4763	0.5117	0.4718
Coverage↓	**4.2243**	4.7887	5.3682	3.050	4.4921
Ranking-loss↓	0.0883	0.1005	0.1127	**0.0770**	0.093
Average-precision↑	**0.6410**	0.6279	0.6211	0.6083	0.6331

Table 9. Average test results of five multi-label algorithms on entertainment dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0525	0.0523	**0.0515**	0.0574	0.0525
One-error↓	**0.3990**	0.4101	0.4067	0.4977	0.4130
Coverage↓	**2.9200**	3.1524	3.3899	2.9623	3.3208
Ranking-loss↓	**0.0984**	0.1085	0.1178	0.1060	0.1139
Average-precision↑	**0.6913**	0.6811	0.6794	0.6255	0.6778

Table 10. Average test results of five multi-label algorithms on health dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0337	**0.03290**	0.0335	0.0363	0.0347
One-error↓	0.2573	**0.2545**	0.2602	0.3043	0.2716
Coverage↓	3.1807	3.5523	3.7748	**2.7877**	3.5983
Ranking-loss↓	0.0489	0.0557	0.0587	**0.0473**	0.0563
Average-precision↑	**0.7888**	0.7866	0.7812	0.7560	0.7775

Table 11. Average test results of five multi-label algorithms on recreation dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0557	**0.0547**	**0.0547**	0.0584	0.0565
One-error↓	**0.4577**	0.4721	0.4712	0.5557	0.4684
Coverage↓	**4.140**	4.4190	4.5297	4.2827	4.2339
Ranking-loss↓	**0.1397**	0.1514	0.1549	0.1535	0.1436
Average-precision↑	**0.6354**	0.6212	0.6201	0.5677	0.6269

Table 12. Average test results of five multi-label algorithms on reference dataset using 10 runs of 5-CV.

Evaluation criterion	**Algorithm**				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0254	**0.0251**	0.0252	0.0270	0.0257
One-error↓	**0.3583**	0.3636	0.3648	0.4043	0.3762
Coverage↓	3.0867	3.5390	3.9425	**2.7340**	3.7601
Ranking-loss↓	0.0717	0.0824	0.0929	**0.0678**	0.0889
Average-precision↑	**0.7217**	0.7134	0.7069	0.6886	0.7050

Table 13. Average test results of five multi-label algorithms on science dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0316	0.0309	0.0308	0.0334	0.0312
One-error↓	**0.5027**	0.5046	0.5061	0.5590	0.4992
Coverage↓	5.9303	7.1455	7.2828	**5.7890**	6.9849
Ranking-loss↓	**0.1079**	0.1330	0.1362	0.1107	0.1300
Average-precision↑	**0.6000**	0.5854	0.5830	0.5483	0.5880

Table 14. Average test results of five multi-label algorithms on social dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	0.0208	**0.0203**	0.0206	0.0126	0.0206
One-error↓	0.2887	0.2849	**0.2819**	0.3133	0.2853
Coverage↓	3.5127	3.8253	4.3701	**2.9770**	4.0361
Ranking-loss↓	0.0608	0.0664	0.0770	**0.0546**	0.0709
Average-precision↑	**0.7715**	0.7685	0.7644	0.7568	0.7664

on yeast dataset, ML-CK-ELM has achieved the best performance except in the one-error evaluation criteria. On scene dataset, Table 3 indicates that ML-CK-ELM performs much better than all the other algorithms in terms of all evaluation metrics. Table 4, on delicious dataset, shows that ML-CK-ELM is inferior to ML-ELM-RBF only in terms of one error and average precision.

From the results shown in Table 5, on art dataset, ML-CK-ELM achieves impressive performance except in terms of hamming loss. As seen in Table 6, on business dataset, the proposed method can reach satisfactory results in hamming loss and average precision metrics. On computer dataset in Table 8, ML-CK-ELM achieves better performance in terms of one error and average precision. Table 8, on education dataset, shows that ML-CK-ELM reaches acceptable results in terms of all evaluation metrics, except ranking-loss. According to Tables 9 and 11 on entertainment dataset and recreation dataset, respectively, ML-CK-ELM has superior performance in terms of one error, coverage, ranking loss, and average precision. On health and social datasets presented in Tables 10 and 14, respectively, ML-CK-ELM has the optimal indicator of average precision. Nonetheless, it performs well in other criteria on the second rank. From the results in Table 12, on reference dataset, ML-CK-ELM is inferior to ML-ELM-RBF and ML-KNN in terms of hamming loss, coverage, and ranking loss. As seen from Table 13, on science dataset, the proposed method obviously achieves superior results in terms of one error, ranking loss, and average precision. Eventually, from the result in Table 15, on society dataset, ML-CK-ELM achieves a satisfactory result in terms of all evaluation criteria, except coverage.

For the sake of making a fair and comprehensive comparison between the proposed method and the other methods, non-parametric statistical analyses based on the Freidman tests are given in the following. The Friedman test [21] is performed on the average value of each metric over all datasets to rank the algorithms and after that, post hoc is used to confirm

Table 15. Average test results of five multi-label algorithms on society dataset using 10 runs of 5-CV.

Evaluation criterion	Algorithm				
	ML-CK-ELM	**ML-ELM-RBF**	**ML-RBF**	**ML-KNN**	**RELM**
Hamming-loss↓	**0.0507**	0.0512	0.0515	0.0535	0.0519
One-error↓	**0.3970**	0.3971	0.4045	0.4247	0.4007
Coverage↓	5.5473	6.0138	6.3341	**5.3110**	6.2731
Ranking-loss↓	**0.1288**	0.1416	0.1491	0.1318	0.1481
Average-precision↑	**0.6419**	0.6345	0.6274	0.6181	0.6304

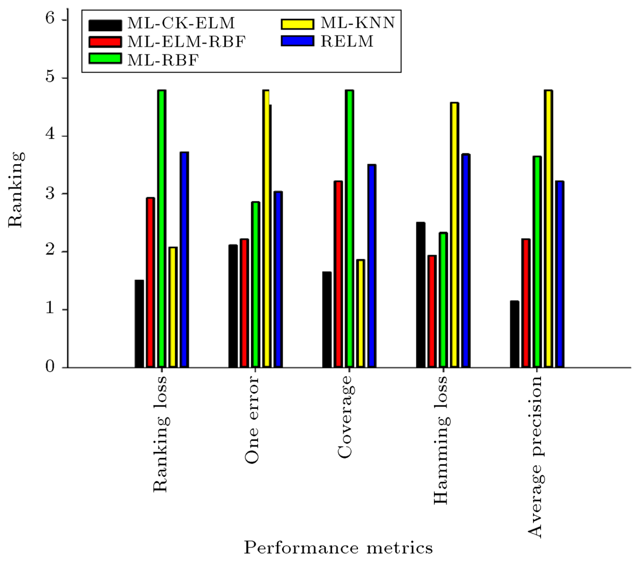

Figure 6. Average ranks obtained by each method on different performance metrics in Friedman test over all datasets (the lower rank is the better performance).

where the differences between algorithms occurred. According to Figure 6, ML-CK-ELM is ranked the first in terms of ranking loss, one error, coverage, and average precision and it can be selected as a control classifier. As shown in Table 16 for ranking-loss, the ML-CK-ELM is compared with the other methods and the Holm's procedure rejects all default hypotheses since the corresponding p-values are less than or equal to 0.05. Therefore, all algorithms are significantly different from ML-CK-ELM on ranking loss metric. In Table 17, the comparison for one error metric is presented, and Holm's procedure rejects the first two hypotheses since the corresponding p-values ≤ 0.016667. Therefore, ML-KNN and RELM are significantly different from ML-CK-ELM in terms of one error. In Table 18, on coverage measure, Holm's procedure rejects all default hypotheses because the corresponding p-values are less than or equal to 0.05. Thus, ML-CK-ELM outperforms ML-RBF, RELM, ML-ELM-RBF, and ML-KNN significantly. Regarding the average-precision results presented in Table 19, the Holm's procedure confirms the superiority of ML-CK-ELM to ML-KNN, ML-RBF, RELM, and ML-ELM-RBF. However, on hamming loss metric, the proposed method is given the third rank, ML-RBF the second, and the ML-ELM-RBF the first, as shown in Figure 6. According to Holm's procedure results of the hamming loss given in Table 20, those default hypotheses whose corresponding p-values are lower than or equal to 0.025 are rejected. Therefore, in terms of hamming loss, ML-ELM-RBF is better than ML-CK-ELM.

According to the post-hoc procedures, our proposed ML-CK-ELM is superior to other methods in

Table 16. Post Hoc comparisons for $\alpha = 0.05$ on ranking-loss; Holm's p-value = 0.05. Control method is Multi-Layer Combined Kernel Extreme Learning Machine (ML-CK-ELM).

Algorithm	Z	p	Holm
ML-RBF	5.49852	0	0.0125
RELM	3.705209	0.000211	0.016667
ML-ELM-RBF	2.390457	0.016827	0.025
ML-KNN	0.956183	0.33898	0.05

Table 17. Post Hoc comparisons for $\alpha = 0.05$ on one-error; Holm's p-value = 0.016667. Control method is Multi-Layer Combined Kernel Extreme Learning Machine (ML-CK-ELM).

Algorithm	Z	p	Holm
ML-KNN	4.482107	0.000007	0.0125
RELM	1.553797	0.120233	0.016667
ML-RBF	1.25499	0.209482	0.025
ML-ELM-RBF	0.179284	0.857714	0.05

Table 18. Post Hoc comparisons for $\alpha = 0.05$ on coverage; Holm's p-value = 0.05. Control method is Multi-Layer Combined Kernel Extreme Learning Machine (ML-CK-ELM).

Algorithm	Z	p	Holm
ML-RBF	5.259006	0	0.0125
RELM	3.107594	0.001886	0.016667
ML-ELM-RBF	2.629503	0.008551	0.025
ML-KNN	0.358569	0.719918	0.05

Table 19. Post Hoc comparisons for $\alpha = 0.05$ on average-precision; Holm's p-value = 0.05. Control method is Multi-Layer Combined Kernel Extreme Learning Machine (ML-CK-ELM).

Algorithm	Z	p	Holm
ML-KNN	6.095666	0	0.0125
ML-RBF	4.1833	0.000029	0.016667
RELM	3.466163	0.000528	0.025
ML-ELM-RBF	1.792843	0.072998	0.05

Table 20. Post Hoc comparisons for $\alpha = 0.05$ on hamming loss; Holm's p-value = 0.025. Control method is Multi-Layer Extreme Learning Machine-Radial Basis Function (ML-ELM-RBF).

Algorithm	Z	p	Holm
ML-KNN	4.422346	0.00001	0.0125
RELM	2.92831	0.003408	0.016667
ML-CK-ELM	0.956183	0.33898	0.025
ML-RBF	0.657376	0.5109939	0.05

terms of average precision, coverage, and ranking loss criteria. To be more specific, due to the average precision considered in parameters tuning via cross validation, ML-CK-ELM has better results in terms of this criterion. Although ML-CK-ELM seems to be better than the other methods according to Figure 6 in terms of one error, there is no significant difference among ML-CK-ELM, ML-RBF, and ML-ELM-RBF. It is worth noting that ML-CK-ELM has a worse performance than ML-ELM-RBF in terms of hamming loss.

In conclusion, the proposed method outperforms the other algorithms in almost all cases in terms of four evaluation criteria, namely average precision, one error, ranking loss, and coverage and it is not better than ML-ELM-RBF only for hamming loss measure. Obtained results show quite an acceptable performance of ML-CK-ELM compared to the other methods. For more illustration, some radar charts are given in Figure 7. In order to make a fair comparison between radar charts, we first normalized the coverage measure by dividing

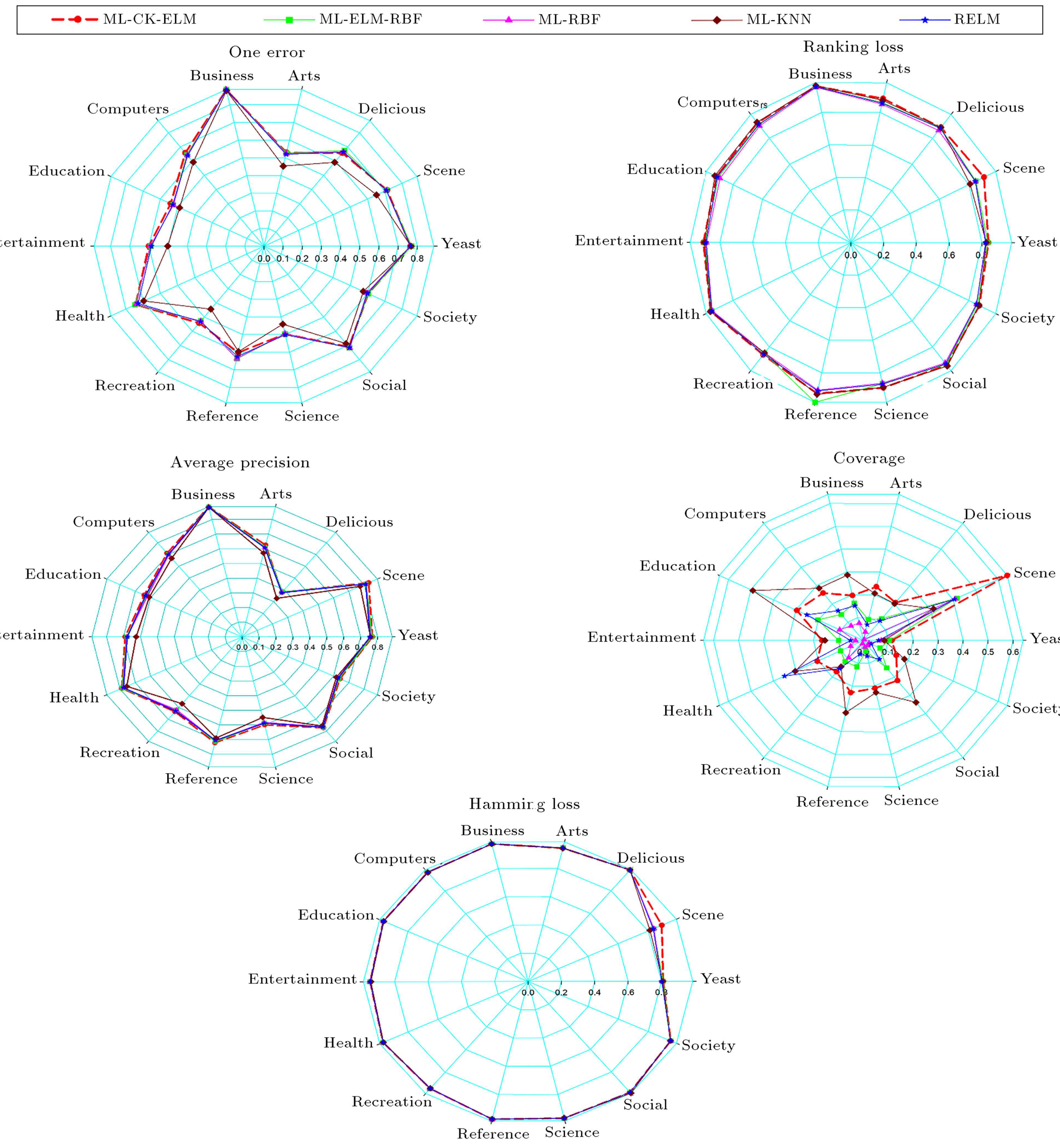

Figure 7. Radar charts for average precision and newly defined criteria of ranking loss, one error, coverage, and hamming loss. The larger the area of radar charts for a method means the superiority of that method to all datasets.

its values into maximum values for each dataset separately to scale them in the interval [0 1]. Then, the reverse values of four criteria ranking including loss, one error, coverage, and hamming loss are obtained via (1-normalized value), showing the higher values of these new measures to be better. Accordingly, the radar chart for each new measure should have a larger area. Based on Figure 7, the dashed line indicates the radar chart of our proposed method that has the maximum area, compared to the other methods. These radar charts visually depict the performance of ML-CK-ELM.

5. Concluding remarks

In this paper, an efficient multi-label classification method based on kernel-based deep learning algorithm called ML-CK-ELM was proposed, which resolved several practical issues of Multi-Layer Extreme Learning Machine (ML-ELM). First, due to kernel-based learning, no random parameter adjustment is necessary. Next, there is no need for adjusting any manual parameter. Finally, both computation time and memory storage were significantly reduced. Furthermore, when some base kernels were combined, an exceptional ability of global approximation was represented, while it showed a good generalized performance. The experimental results indicated that the proposed multi-label classification method enjoyed extremely enhanced capability of multi-label classification compared with state-of-the-art algorithms. In brief, ML-CK-ELM made a satisfactory improvement on the multi-label classification task over state-of-the-art algorithms. As a framework of further investigations, it is recommended that an extensive research be conducted by employing different ways of kernel combination such as non-linear combinations and their advantages over the linear combination.

References

1. Charte, F., del Jesus, M.J., and Rivera, A.J. "Multilabel classification: Problem analysis, metrics and techniques", In *Multilabel Classification: Problem Analysis, Metrics and Techniques*, Springer (2016).
2. Mercan, C., Aksoy, S., Mercan, E., Shapiro, L.G., Weaver, D.L., and Elmore, J.G. "Multi-instance multi-label learning for multi-class classification of whole slide breast histopathology images", *IEEE Transactions on Medical Imaging*, **37**(1), pp. 316–325 (2017).
3. Gibaja, E. and Ventura, S. "Multi-label learning: a review of the state of the art and ongoing research", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **4**(6), pp. 411–444 (2014).
4. Zhang, M.-L. and Zhou, Z.-H. "A review on multi-label learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, **26**(8), pp. 1819–1837 (2014).
5. Wu, Y.-P. and Lin, H.-T. "Progressive random k-labelsets for cost-sensitive multi-label classification", *Machine Learning*, **106**(5), pp. 671–694 (2017).
6. Trajdos, P. and Kurzynski, M. "Dynamic classifier chains for multi-label learning", In *German Conference on Pattern Recognition*, pp. 567–580 (2019).
7. Zhang, M.-L. and Zhou, Z.-H. "ML-KNN: A lazy learning approach to multi-label learning", *Pattern Recognition*, **40**(7), pp. 2038–2048 (2007).
8. Zhang, M.-L. "M l-rbf: Rbf neural networks for multi-label learning", *Neural Processing Letters*, **29**(2), pp. 61–74 (2009).
9. Deng, W.-Y., Zheng, Q.-H., Chen, L., and Xu, X.-B. "Research on extreme learning of neural networks", *Chinese Journal of Computers*, **33**(2), pp. 279–287 (2010).
10. Zhang, N., Ding, S., and Zhang, J. "Multi layer ELM-RBF for multi-label learning", *Applied Soft Computing*, **43**, pp. 535–545 (2016).
11. Kasun, L.L.C., Yang, Y., Huang, G.-B., and Zhang, Z. "Dimension reduction with extreme learning machine", *IEEE Transactions on Image Processing*, **25**(8), pp. 3906–3918 (2016).
12. Kasun, L.L.C., Zhou, H., Huang, G.-B., and Vong, C.M. "Representational learning with extreme learning machine for big data", *IEEE Intelligent Systems*, **28**(6), pp. 31–34 (2013).
13. Tissera, M.D. and McDonnell, M.D. "Deep extreme learning machines for classification", In *Proceedings of ELM-2014*, **1**, pp. 345–354, Springer (2016).
14. Tang, J., Deng, C., and Huang, G.-B. "Extreme learning machine for multilayer perceptron", *IEEE Transactions on Neural Networks and Learning Systems*, **27**(4), pp. 809–821 (2016).
15. Huang, G.-B., Zhou, H., Ding, X., and Zhang, R. "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **42**(2), pp. 513–529 (2012).
16. Wong, C.M., Vong, C.M., Wong, P.K., and Cao, J. "Kernel-based multilayer extreme learning machines for representation learning", *IEEE Transactions on Neural Networks and Learning Systems*, **29**(3), pp. 757–762 (2018).
17. Chen, T., Wang, S., and Chen, S. "Deep multimodal network for multi-label classification", In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 955–960 (2017).
18. Huang, Y., Wang, W., and Wang, L. "Unconstrained multimodal multi-label learning", *IEEE Transactions on Multimedia*, **17**(11), pp. 1923–1935 (2015).

19. Li, K., Zou, C., Bu, S., Liang, Y., Zhang, J., and Gong, M. "Multi-modal feature fusion for geographic image annotation", *Pattern Recognition*, **73**, pp. 1–14 (2018).

20. Song, L., Liu, J., Qian, B., Sun, M., Yang, K., Sun, M., et al. "A deep multi-modal CNN for multi-instance multi-label image classification", *IEEE Transactions on Image Processing*, **27**(12), pp. 6025–6038 (2018).

21. García, S., Fernández, A., Luengo, J., and Herrera, F. "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power", *Information Sciences*, **180**(10), pp. 2044–2064 (2010).

Biographies

Mohammad Rezaei Ravari has been an MSc student of Artificial Intelligence in Engineering, Shahid Bahonar of University, Kerman, Iran since 2017. His research interests include machine learning, deep learning, and multi-classifier systems.

Mahdi Eftekhari has been a faculty member of Shahid Bahonar University of Kerman, Kerman, Iran since 2008 and he is currently an Associate Professor at the Department of Computer Engineering. His research interests include data mining, fuzzy systems and modeling, machine learning, deep learning, and applications of matrix methods in machine learning. He is the author and co-author of about 120 papers in cited journals and conferences.

Farid Saberi-Movahed works as a Research Assistant at the Department of Applied Mathematics, Faculty of Sciences and Modern Technologies, Graduate University of Advanced Technology, Kerman, Iran. Since 2017, he has made significant contributions at the Department of Computer Engineering, Shahid Bahonar University of Kerman. His research area includes numerical linear algebra, tensor computation, machine learning, and feature selection.