Multi-Label classifier based on Kernel Random Vector Functional Link Network

1st Vikas Chauhan
Discipline of Computer Science and Engineering
Indian Institute of Technology
Indore, India
phd1701101006@iiti.ac.in

2nd Aruna Tiwari
Discipline of Computer Science and Engineering
Indian Institute of Technology
Indore, India
artiwari@iiti.ac.in

3rd Shivvrat Arya
Department of Computer Science and Engineering
The University of Texas at Dallas
Dallas,USA
Shivvrat.Arya@utdallas.edu

***Abstract*—In this paper, a kernelized version of the random vector functional link network is proposed for multi-label classification. This classifier uses pseudoinverse to find output weights of the network. As pseudoinverse is non-iterative in nature, it requires less fine-tuning to train the network. Kernelization of RVFL makes it robust and stable as no need to tune the number of neuron in the enhancement layer. A threshold function is used with a kernelized random vector functional link network to make it suitable for multi-label learning problems. Experiments performed on three benchmark multi-label datasets bibtex, emotions, and scene shows that proposed classifier outperforms various the existing multi-label classifiers.**

***Index Terms*—Random vector functional link, multi-label classification, kernel random vector functional link, non-iterative neural networks, pseudoinverse.**

I. Introduction

Traditional classification problems comprise of assigning a single class or label to the data instance from two (binary classification) or more than two(multi-class classification) labels. But in a real-life scenario, a single instance can be assigned multiple labels. For example, in a text classification problem, a single instance can be assigned multiple labels such as sports, cricket and world cup [1], [2]. Multi-label classification problems are a more universal and generalized version of the multi-class classification problem. But this generalization brings more challenges and difficulties to the problem as the size of the output spaces increases exponentially. Therefore, multi-label classification is a fast-growing field of research in recent time. The origin of multi-label problem is stared with text classification [1]–[4]. Some other applications of multi-label classification are genomics and image processing [5], [6] are arises in the recent time.

Multi-label classification algorithms are broadly categorized into two categories as Problem Transformation and Algorithms Adaptation [7]. Problem Transformation transforms multi-label classification to multiple independent single label classifier. Binary Relevance(BR), Classifier Chain(CC), and Label Power Set(LP) are the main algorithms under the category of Problem Transformation. The BR method converts multi-class classification to independent binary classifier and trains for each label independently [5]. For Q output labels, BR converts the multi-label problem to Q independent multi-class problems. Each of the binary classifiers votes separately for the final result. A new class for each combination of output labels are generated by Label Powerset (LP). Then these combinations are solved as a multi-class classification in the LP algorithm. In the LP algorithm, the number of classes is increased in exponential growth, and this may lead to overfitting [8]. The correlation among the output labels is not considered in BR; this shortcoming of BR is overcome by LP and known as Label Cardinality. Classifier Chain (CC) is another problem transformation approach which performs better than BR classifier [9]. In CC, the output of the previous binary classifier is also provided as an input to next classifier with the input data, and this is the difference with BR. The shortcoming of CC is that its performance depends on the order of output labels.

Algorithm adaptation adapts the existing multi-class and binary class algorithms to learn multi-label classification. A neural network based multi-label classification algorithm Back-Propagation Multi-Label Learning(BPMLL) was derived from the popular Backpropagation algorithm [6]. BP-MLL replaced the error function of backpropagation with different function which was able to learn the multi-label characteristics. The main aim of this change was to create a condition in which the labels belonging to a label ranked higher than those not belonging to that label. As backpropagation is an iterative procedure, its convergence depends on the iterations. Rank-SVM is Support Vector Machine (SVM) based multi-label classifier [11]. It learns the multi-label

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

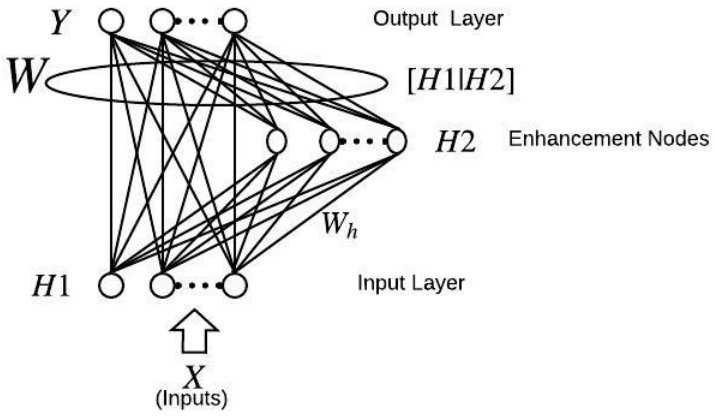

Fig. 1. RVFL structure [10]

problem by constructing SVM for each output label according to the ranking loss. It leads to the learning of correlation among the labels. Random Vector Functional Link (RVFL) Network is a single layer feedforward neural network which follows non-iterative learning. The output weights are computed by pseudoinverse in RVFL, and it makes it faster to train. We propose kernelized RVFL for multi-label classification which is adapted from the kernelized RVFL. Extreme Learning Machine (ELM) is similar to RVFL without having direct links from input to output layer [12]. This is the main difference between ELM and RVFL. ML-KELM uses the Support Vector Machine based adaptive threshold, which increases its training time because it depends on the number of output labels. In case of a large number of output labels, ML-KELM becomes very time consuming to converge. The direct links between the input and output layer make the performance of RVFL better than ELM, as shown in figure 1 in 1 [13]. Kernelization of RVFL [14] provides the stabilization to the network.

We propose multi-label classifier based on Kernel Random Vector Functional Link Network (ML-KRVFL) which is an adaptation of kernelize RVFL for multi-label classification. As K-RVFL is not used for multi-label classification, we adapt the RVFL using kernel function for multi-label problems. We have used the simple threshold mechanism which takes the average value of the gap of output values provided by the ML-KRVFL. Section II describes the preliminaries for non-iterative learning and Kernel RVFL. Section III describes the proposed multi-label classifier based on kernel Multi-label Random Vector Functional Link Network followed by the discussion of experiments and result in section V. In the last section VI, the conclusion of the proposed ML-KRVFLN approach is given.

II. Related Work

RVFL uses the pseudoinverse to compute the weights of neural network. In this section, The pseudoinverse with RVFL and formal definition multi-label classification is discussed to provide the basic of Kernelized RVFL. The mathematical formulation of RVFL network and pseudo inverse is discussed in this section to make preliminaries for proposed ML-KRVFL.

A. Multi-Label classification

The aim of multi-label classifier $h : X \to 2^{\mathcal{Y}}$ is to optimize some specific evaluation metric. The learning system produces a real valued function $f : X \times \mathcal{Y} \to \mathbb{R}$. For a given instance $x_i \subseteq X$ and its associated label set $Y_i \subseteq \mathcal{Y}$, these real valued systems are supposed to output the larger values for labels Y_i than the labels not in Y_i. For any $y_1 \in Y_i$ and $y_2 \notin Y_i$, These real values functions are denoted as $f(x_i, y_1) > f(x_i, y_2)$ and can be transformed as the ranking function $rank_f(.,.)$. The formal mathematical definition of multi-label classification is as follows:

1) Let $X \in \mathbb{R}^{\mathrm{M}}$ represents a M-dimensional instance space, and $\mathcal{Y} = \{y_1, y_2, ..., y_q\}$ denotes the label space with q possible class labels,where $q > 1$.
2) Multi-label learning learns a functions $f : X \times \mathcal{Y} \to \mathbb{R}$ from the multi-label training set $T = \{(x_i, Y_i)|1 \leq i \leq N\}$.
3) For each multi-label training example (x_i, y_i), $x_i \in X$ is a M-dimensional feature vector $x_i = (x_{i1}, x_{i2}, ..., x_{iM})^T$ and $Y_i \subseteq \mathcal{Y}$.
4) For any unseen instance $x \in X$, the multi-label classifier $h(.)$ predicts $h(x) \subseteq \mathcal{Y}$ as the set of proper labels for x.

B. Random Vector Functional Link Network

Pao and Takefuji has proposed a random vector functional link network, as represented in figure 1. It shows that weight

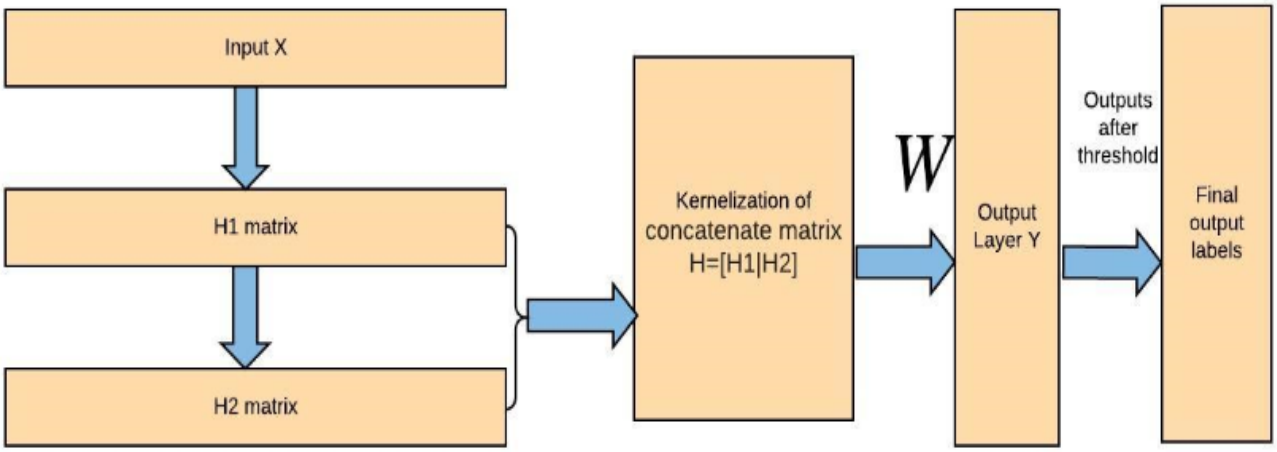

Fig. 2. Structure of multi-label kernal random vector functional-link Neural Network

values can be generated randomly between enhancement node layers and input layer. The layer of Enhancement nodes denoted as $H2$ and input layer $H1$ are concatenated as shown in Fig. 1. The output weight W is computed by using pseudoinverse [15]. As shown in figure 1 input data represented by matrix X with size $N \times M$ is provided to input layer $H1$. N denotes the input samples and M denotes the dimensions of each input sample. W_h are the weights which are generated randomly. Enhancements nodes are generated by the multiplication of W_h and input layer $H1$. The concatenation of $H1$ and $H2$ can be shown as follows:

$$H = [H1|H2] \tag{1}$$

As at outputs at the output layer Y are known during the training time, The pseudoinverse can be used to find the output weights W. The RVFL network can be shown by the following equation:

$$HW = Y \tag{2}$$

Pseudoinverse [15] is the noniterative method to compute the output weights W on RVFL network as shown in Fig. 1. Lets $S(W)$ is the error and it is a function of$W$. The aim is to minimise the difference between Y and HW and also to reduce the weight values.

$$S(W) = \frac{1}{2}\left\|Y - HW\right\|_2^2 + \frac{1}{2C}\left\|W\right\|_2^2 \tag{3}$$

A positive value C is used for regularization and to avoid the singularity. We can rewrite (3) as follows:

$$S(W) = \frac{1}{2}\left(Y - HW\right)^T\left(Y - HW\right) + \frac{1}{2C}W^TW \tag{4}$$

The error $S(W)$ represented in (4) can be written as follows:

$$S(W) = \frac{1}{2}\left(Y^T - W^TH^T\right)\left(Y - HW\right) + \frac{1}{2C}W^TW \tag{5}$$

Further the error $S(W)$ shown in (4) can be simplified as follows:

$$S(W) = \frac{1}{2}\left(Y^TY - Y^THW - W^TH^TY + W^TH^THW\right)^T + \frac{1}{2C}W^TW \tag{6}$$

Differentiating (6) with respect to W, we get the following equation:

$$\begin{aligned}\frac{dS(W)}{dW} &= \frac{1}{2}\left(-H^TY - H^TY + 2H^THW\right) + \frac{1}{C}(W)\\ &= -H^TY + H^THW + \frac{1}{C}(W)\end{aligned} \tag{7}$$

At minima of $\frac{dS(W)}{dW}$ = 0, substituting in (7), we get the following equation:

$$\begin{aligned}&-H^TY + \left(H^TH + \frac{1}{C}\right)W = 0\\ &\text{or}\\ &\left(H^TH + \frac{1}{C}\right)W = H^TY\end{aligned} \tag{8}$$

Now we can get the output layer weights W as shown in following equation:

$$\begin{aligned}W &= \left(H^TH + \frac{1}{C}\right)^{-1}H^TY\\ W &= H^+Y\end{aligned} \tag{9}$$

Now , we have pseudoinverse from (9) in following Eq.

$$H^+ = \left(H^TH + \frac{1}{C}\right)^{-1}H^T \tag{10}$$

or above equation can written as follows:

$$H^+ = H^T\left(H^TH + \frac{I}{C}\right)^{-1} \tag{11}$$

This H^+ is known as pseudoinverse and it is used for the computation of output layer weights W as shown in figure 1 of RFVLN.

III. Proposed Work

In this section, we discuss the ML-KRVFL that has been proposed to solve the multi-label classification. This is a single layer feedforward neural network and a modification of RVFL for multi-label classification. In ML-KRVFL, a Kernel function is used in place of the enhancement layer. This kernelization provides the robustness to the RVFL and reduces the randomness effect of the enhancement layer. It is reduced because in ML-KRVFL activation function of enhancement nodes and the number of enhancement nodes in enhancement layer $H2$ need not be known. We discuss the formulation and structure of ML-KRVFL in this section. Using the RVFL network from (1) and (2), the raw output at the layer Y of RVFL can be written as:

$$f(x) = HW \tag{12}$$

by substituting the value of W from (9) to (12), $f(x)$ can be written as:

$$f(x) = HH^T \left(H^T H + \frac{I}{C} \right)^{-1} Y \tag{13}$$

by substituting the value of H from (1) in (13), the output $f(x)$ of ML-KEVFL can be represented as follows:

$$f(x) = [H1|H2] \left[\frac{H1}{H2} \right]^T \left([H1|H2] \left[\frac{H1}{H2} \right]^T + \frac{I}{C} \right)^{-1} Y \tag{14}$$

Then (14) can be rewritten as follows:

$$\begin{aligned} f(x) &= \left[H1|H1^T \right] + \\ &\left[H2|H2^T \right] \left(\left[H1|H1^T \right] + \left[H1|H2^T \right]^T + \tfrac{I}{C} \right)^{-1} Y \end{aligned} \tag{15}$$

The kernel matrix for the (15) can be defined as follows:

$$\Omega = H1H1^T = K(x, x^T) \tag{16}$$

$$\tilde{\Omega} = H2H2^T = \tilde{K}(x, x^T) \tag{17}$$

$$\begin{bmatrix} K(x, x_1) \\ . \\ . \\ . \\ K(x, x_L) \end{bmatrix}^T + \begin{bmatrix} \tilde{K}(x, x_1) \\ . \\ . \\ . \\ \tilde{K}(x, x_L) \end{bmatrix}^T \left(\Omega + \tilde{\Omega} + \frac{I}{C} \right)^{-1} Y \tag{18}$$

K is a linear kernel and $\tilde{K}$ is the radial basis function used in this paper.

For the raw output $f(x)$ of ML-KRVFL at Y, there should be a threshold function for conversion of these raw values of $f(x)$ to a binary vector of 1 or -1 for each input data sample. Here 1 denotes that output label belongs to input data instance and -1 denotes that output label does not belong to input data instance. After the completion training, the values of $f(x)$ for each label of all instances can be divided into two sets first those values which belong to labels Y_1 another which does not belong to label Y_1. The values which do not belong to Y_1, have been represented by Y_2. For each label, threshold values can be computed as follows:

$$\text{Threshold for each label} = \frac{min(Y_1) + max(Y_2)}{2} \tag{19}$$

At the test time, a test sample is passed through ML-KRVFL network, and $f(x)$ provides the output values at the output layer Y. These values are compared with threshold values, and final output labels can be found as follows:

$$\text{final labels corresponds to } Y_i = \begin{cases} 1, & f_k(x_i) \geq threshold \\ -1, & f_k(x_i) < threshold \end{cases} \tag{20}$$

where, $i = 1, ..., N^{'}$. and $k = 1, ..., Q$. Total no. of test samples are denoted by $N^{'}$ in Eq. (20) The complete algorithm for multi-label classification using ML-RVFL is described in algorithm 1.

TABLE I
Multi-label data-set description

Data	Attributes	Examples	Labels
bibtex	1836	7395	159
emotions	72	593	6
scene	294	2407	6

Algorithm 1: Multi-label kernel Random Vector Functional Link Neural Network

Input : training set and testing set
Output: W (output weight matrix) and threshold

Initialize kernel parameters and regularization cost parameter C
Pre-process: Conversion of output label values of training set to bipolar values 1 or -1;
for `training set` **do**
  Compute kernel matrix Ω and $\tilde{\Omega}$
  Computation of network using (18)
  Computation of threshold using (19)
end
for `testing set` **do**
  Using output weight matrix W computed by (9), compute raw output values of ML-KRVFL network $f(x)$ using (18)
  Compare ML-KRVFL network output $f(x)$ with a threshold value and obtain multi-label identification final outputs according to (20)
end

IV. Evaluation metrics

This section discusses the evaluation metrics that are used to compare the results of the proposed ML-KRVFL algorithm with the existing state of the art multi-label classification algorithms. In multi-label classification, five evaluation measures hamming loss, one error, coverage,

TABLE II
COMPARISON RESULTS ON BIBTEX DATASET

Algorithms → Evaluation Metrics ↓	**Rank - SVM**	**ML-ELM**	**BPMLL**	**ML-KELM**	**ML-KRVFL**
Hamming Loss	0.0196±0.0002	0.0148±0.0003	0.0162±0.0003	0.0151±0.0001	**0.0145 ± 0.0005**
Ranking loss	0.329±0.0061	0.6687±0.0194	**0.0696±0.0039**	0.4187±0.0099	0.1060 ± 0.0083
One error	0.859±0.0035	**0.1847±0.0041**	0.5384±0.0313	0.6038±0.0118	0.3601 ± 0.011
Coverage	0.774±0.0203	0.6289±0.0213	**0.0985±0.0038**	0.3516±0.0075	0.2023± 0.012
Avg precision	0.139±0.0013	0.3685±0.015	0.548±0.0091	0.1145±0.0064	**0.5744 ± 0.009**

TABLE III
COMPARISON RESULTS ON EMOTIONS DATASET

Algorithms → Evaluation Metrics ↓	**Rank - SVM**	**ML-ELM**	**BPMLL**	**ML-KELM**	**ML-KRVFL**
Hamming Loss	0.3297±0.017	0.3607±0.038	0.3213±0.0144	0.6793±0.0095	**0.2156 ± 0.0192**
Ranking loss	0.4151±0.0244	0.9736±0.0042	0.4723±0.0247	0.3483±0.0165	**0.1760 ± 0.0292**
One error	0.5548±0.0598	0.5124±0.0208	0.6667±0.2887	0.5333±0.2173	**0.2715± 0.0235**
Coverage	**0.3127±0.2145**	0.4667±0.1826	0.5494±0.0236	0.4213±0.0246	0.3142 ± 0.0338
Avg precision	0.572±0.0245	0.0992±0.0454	0.5583±0.0187	0.4735±0.0268	**0.7947 ± 0.024**

TABLE IV
COMPARISON RESULTS ON SCENE DATASET

Algorithms → Evaluation Metrics ↓	**Rank - SVM**	**ML-ELM**	**BPMLL**	**ML-KELM**	**ML-KRVFL**
Hamming Loss	0.2718±0.0082	0.1545±0.0741	0.2839±0.0067	0.767±0.0465	**0.119 ± 0.0050**
Ranking loss	0.4844±0.0282	0.4509±0.0948	0.3159±0.0411	0.1675±0.0138	**0.1187 ± 0.0107**
One error	0.7786±0.0234	0.2667±0.117	0.6±0.1491	**0.0667±0.0913**	0.2995 ± 0.0194
Coverage	0.24807±0.096	0.7605±0.3028	0.3628±0.0124	**0.0889±0.0094**	0.1146 ± 0.010
Avg precision	0.4396±0.0173	0.1645±0.2676	0.4526±0.0074	0.6733±0.0278	**0.8126 ± 0.014**

TABLE V
AVERAGE PERFORMANCE ON BIBTEX, EMOTIONS, AND SCENE DATASETS

Algorithms → Evaluation Metrics ↓	**Rank - SVM**	**ML-ELM**	**BPMLL**	**ML-KELM**	**ML-KRVFL**
hamming loss	0.207±0.0085	0.1767±0.0375	0.2071±0.0071	0.4871±0.0187	**0.1164±0.0082**
Ranking Loss	0.4095±0.0196	0.6977±0.0395	0.2859±0.0232	0.3115±0.0134	**0.1336±0.0161**
One Error	0.7308±0.0289	0.3213±0.0473	0.6017±0.1564	0.4013±0.1068	**0.3104±0.0182**
Coverage	0.445±0.1103	0.6187±0.1689	0.3369±0.0133	0.2873±0.0138	**0.2104±0.0186**
Avg precision	0.3835±0.0144	0.2107±0.1093	0.5196±0.0117	0.4204±0.0203	**0.7272±0.0161**

ranking loss, and average precision are used for the comparison among the algorithms [4]. For a given test set $T^{'} = \{(x_1, Y_1), (x_2, Y_2), ..., (x_{N'}, Y_{N'})\}$, the evaluation metrics are discussed below.

A. Hamming loss

Hamming loss is defined as the hamming distance between the predicted value and the actual result, i.e. it counts for the number of instances where the predicted values are not equal to the actual result. The smaller the value of the hamming loss, the better the performance of the algorithm.

$$\text{hamming loss} = \frac{1}{N'} \sum_{i=1}^{N^{'}} \frac{1}{Q} |h(x_i) \triangle Y_i|, \tag{21}$$

where $\triangle$ is the symmetric difference between predicted output and actual output. $h(x_i)$ denotes the predicted output of multi-label classifier and Q is the total number of labels.

B. One error

One error is defined as the number of times the top-ranked label is not present in the set of proper labels of the instance. The smaller the value of the one error, the better the performance of the model. This evaluation metric is defined as

$$\text{one-error} = \frac{1}{N'} \sum_{i=1}^{N^{'}} \left[\left[argmax_{y \in Y} f(X_i, y) \right] \notin Y_i \right] \tag{22}$$

where $f(X_i, y)$ is the raw values provided by multi-label classifier, in ML-KRVFL it is the same as $f(x)$.

C. Coverage

Coverage is defined as the summation of the rank of the most insignificant label which belongs to the instance. In other words it says how many labels we have to check in order to

cover all the proper labels of that instance. The smaller the value of the coverage, the better the performance of the model.

$$\text{coverage} = \frac{1}{N'}\sum_{i=1}^{N'} maxrank_{y\in Y_i}(x_i, y) - 1. \tag{23}$$

As discussed in the mathematicl definition of multi-label classification in section II-A, $rank_f(.,.)$ is generated from the $f(.,.)$. It maps the raw output of for all Q labels such that if $f(x_i, y_2)$ ¡ $f(x_i, y_1)$ in this case the$rank_f(x_i, y_1)$ will be lesser than the $rank_f(x_i, y_2)$

D. Ranking loss

Ranking Loss is defined as the average fraction of label pairs that are reversely ordered for the instance. The smaller the value of the ranking loss , the better the performance of the model.

$$\text{ranking loss} = \frac{1}{N'}\sum_{i=1}^{N'} \frac{1}{|Y_i||\bar{Y_i}|} \tag{24}$$
$$|\{(y_1, y_2)|f(x_i, y_1) \leq f(x_i, y_2), (y_1, y_2 \in Y_i \times \bar{Y_i}\}|$$

where $\bar{Y_i}$ is the complementary set of predicted labels in Y.

E. Average precision

Average Precision is defined as the average fraction of labels ranked above a particular label $y \in Y$ which actually are in Y. The bigger the value of the average precision, the better the performance of the model.

$$\text{average precision} = \frac{1}{N'}\sum_{i=1}^{N'} \frac{1}{|Y_i|}\sum_{y\in Y_i} \frac{|\{y^{'}|rank_f(x_i,y^{'})\leq rank_f(x_i,y), y^{'}\in Y_i\}|}{rank_f(x_i,y)} \tag{25}$$

V. Experiments and Results

In this section, ML-KRVFL has been compared with BP-MLL, Rank-SVM, ML-ELM, and ML-KELM [6], [11], [12], [16] multi-label algorithms. The experimental setup used for comparison and evaluation is the same as mentioned in the papers of Rank-SVM, BP-MLL, ML-ELM, and ML-KELM. For BP-MLL, the number of neurons in hidden layers are set as 20% of input layer neurons, as mentioned in [6]. The 5-fold cross-validation method is used for the experiments after breaking the datasets into the training set and testing set. Same indexing of data for various folds is used for all the algorithms during experimentation. Mean value and standard deviation are shown as a result of all performance measures in table II to IV. All experiments are conducted in MATLAB 2017. The experimentation tests of the proposed kernel RVFL method and the existing state of the art algorithms are performed on a Dell Vostro 3470 SFF Desktop-Core i5 8th Gen computer with 16 GB, DDR4 memory.

Three multi-label benchmark datasets have been used for the experimentation. These datasets are bibtex, emotions, and scene. The details of the datasets are given in table I. The performance of multi-label algorithms for bibtex dataset is given in table II. ML-KRVFL provides optimum hamming loss and average precision. BP-MLL provides the optimum ranking loss and coverage for bibtex and ML-ELM provides the optimum one error.

The performance of multi-label algorithms for emotions dataset is given in table III. ML-KRVFL provides the optimum hamming loss, ranking loss, one error, and average precision. Rank-SVM provides the best coverage, but by observing the values of coverage for ML-KRVFL and Rank-SVM, it is .0015, which is very less.

The performance of multi-label algorithms for scene dataset is given in table IV. ML-KRVFL provides optimum hamming loss, ranking loss, and average precision. ML-KELM provides optimum one error and coverage.

It isn't very easy to provide a single algorithm to provide the optimum result for all evaluation metrics on various datasets, but it can be clearly seen that ML-KRVFL is outperforming the other multi-label algorithms from tables II to table IV. The average of all the evaluation metrics is taken for all the three datasets in table V. This table can be considered as an average rank for all evaluation metrics. The average of all evaluation metrics on three datasets is inspired by average Friedman rank [17]. As three datasets are used for experiments with five classification approaches in this paper, we have used average Friedman rank as an indicative measure in table V. For hamming-loss, ranking loss, one error, and coverage the minimum value denote better rank, and for average precision, large value denotes the better rank, so the average rank of ML-KRVFL is better than Rank-SVM, BP-MLL, ML-ELM, and ML-KELM. It clearly shows that KRVFL is providing the optimum hamming loss, ranking loss, one error, coverage, and average precision as an average of all evaluation metrics. BP-MLL and Rank SVM are iterative methods, and the training of these algorithms is time-consuming. ML-ELM and ML-KELM are non-iterative approaches, and the experimental results show that ML-KRVFL performs better than compared multi-label algorithms.

VI. Conclusion

The purpose of multi-label classification is to assign a set of correct labels for an unseen input sample. In this paper, the ML-KRVFL multi-label classification algorithm is proposed, which adapts kernelized random vector functional link network with a threshold function. The ML-KRVFL algorithm is tested on the benchmark dataset of bibtex, emotions, and scene with the domain of text and image. Experimental results show that ML-KRVFL performs better than existing state-of-the-art algorithms. Additionally, this algorithm is suitable for a large number of input samples as its weights are computed by pseudoinverse. It can be further using incremental learning for fast computation of the weights of ML-KRVFL.

References

[1] G. Salton, "Developments in automatic text retrieval," *science*, vol. 253, no. 5023, pp. 974–980, 1991.

[2] N. Ueda and K. Saito, "Parametric Mixture Models for Multi-labeled Text," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 737–744.

[3] A. K. McCallum, "Multi-label text classification with a mixture model trained by EM," in *AAAI 99 Workshop on Text Learning*, 1999.

[4] R. E. Schapire and Y. Singer, "BoosTexter: A Boosting-based System for Text Categorization," *Machine Learning*, vol. 39, no. 2, pp. 135–168, May 2000.

[5] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757 – 1771, 2004.

[6] Min-Ling Zhang and Zhi-Hua Zhou, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, Oct. 2006.

[7] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819–1837, 2013.

[8] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Machine Learning: ECML 2007*, J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 406–417.

[9] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier Chains for Multi-label Classification," in *Machine Learning and Knowledge Discovery in Databases*, W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 254–269.

[10] C. L. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 1, pp. 62–72, Feb. 1999.

[11] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in neural information processing systems*, 2002, pp. 681–687.

[12] R. Venkatesan, M. J. Er, S. Wu, and M. Pratama, "A novel online real-time classifier for multi-label data streams," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 1833–1840.

[13] L. Zhang and P. N. Suganthan, "A comprehensive evaluation of random vector functional link networks," *Information Sciences*, vol. 367-368, pp. 1094 – 1105, 2016.

[14] K. Xu, H. Li, and H. Yang, "Kernel-based random vector functional-link network for fast learning of spatiotemporal dynamic processes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 1016–1026, May 2019.

[15] C. R. Rao and S. K. Mitra, "Generalized inverse of a matrix and its applications," in *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*. Berkeley, Calif.: University of California Press, 1972, pp. 601–620.

[16] F. Luo, W. Guo, Y. Yu, and G. Chen, "A multi-label classification algorithm based on kernel extreme learning machine," *Neurocomputing*, vol. 260, pp. 313 – 320, 2017.

[17] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, p. 1–30, Dec. 2006.