2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Fast Random k-Labelsets for Large-Scale Multi-Label Classification

Keigo Kimura, Mineichi Kudo, Lu Sun
Graduate School of Information Science and Technology
Hokkaido University, Sapporo, Japan.
{kkimura,mine,sunlu}@main.ist.hokudai.ac.jp

Sadamori Koujaku
Department of Engineering Mathematics
University of Bristol, United Kingdom
sadamori.koujaku@bristol.ac.uk

***Abstract*—Multi-label classification (MLC), allowing instances to have multiple labels, has been received a surge of interests in recent years due to its wide range of applications such as image annotation and document tagging. One of simplest ways to solve MLC problems is label-power set method (LP) that regards all possible label subsets as classes. LP validates traditional multi-classification classifiers such as multi-class SVM but it suffers from the increased number of classes. Therefore, several improvements have been made for LP to be scaled for large problems with many labels. Random k labELsets (RAkEL) proposed by Tsoumakas *et al.* solves this problem by randomly sampling a small number of labels and taking ensemble of them. However, RAkEL needs all instances for constructing each model and thus suffers from high computational complexity. In this paper, we propose a new fast algorithm for RAkEL. First, we assign each training instance to a small number of models. Then LP is applied for each model with only the assigned instances. Experiments on twelve benchmark datasets demonstrated that the proposed algorithm works faster than the conventional methods while keeping accuracy. In the best case, it was 100 times faster than baseline method (LP) and 30 times faster than the original RAkEL.**

I. Introduction

Multi-label classification (MLC) problems, which allow instances to have more than one label at the same time, have become popular because of the natural modeling in several real applications such as image annotation and document tagging. For example, an image of a tiger in the jungle would be labeled "tiger", "tree" and "jungle" at the same time. In such an MLC problem, all possible combinations of labels must be considered, and thus, there are many challenging tasks [1], [2].

There are two extreme ways to solve MLC: binary relevance (BR) and label-power set method (LP) [3]. BR decomposes a MLC problem into a set of independent binary classification problems associated to individual label. In contrast to the simplicity, in BR, the correlation between labels are totally ignored. As a result, its classification accuracy is relatively low [16], [20] LP takes another extreme strategy. It transforms an MLC problem into a multi-class single-label classification problem by considering all possible label subsets as newly extended classes. LP can fully utilize the label correlation but causes an combinatorial explosion instead. Thus, recently, several intermediate methods have been proposed [4]–[6].

Random k-labelsets (RAkEL) is one of such approaches [4]. RAkEL randomly samples some label subsets of a relatively small size and applies LP for each label subset. A test instance is classified by all the classifiers and the results are combined by the majority voting (See Table I). It is known that RAkEL performs better compared to the other MLC methods in many datasets. Usually, the sizes of the label subsets are fixed to a small number to make RAkEL scalable. Nevertheless, RAkEL still suffers from a large computational complexity because it needs all the instances to build each component LP classifier. The reason is as follows. Suppose that a test instance with a hidden label set $\{\lambda_1, \lambda_2, \lambda_3\}$ is classified by five classifiers built on the five corresponding models with label subset $\mathcal{L}^{(m)}$ $(m = 1, 2, \ldots, 5)$ as shown in Table I. Then the expected output of the third classifier associated with $\mathcal{L}^{(3)} = \{\lambda_4, \lambda_5, \lambda_6\}$ is all zero's. However, usually any multi-class classifier is designed to output at least one label, and thus, such "no label" output is not possible. We connect such a "no label" output to a special class and call it *Null class*. RAkEL solves this "no null class problem" by making the null class ϕ explicitly and assigns every instance into one of $2^{|\mathcal{L}^{(m)}|}$ classes, for example, four classes of $\phi, \{1\}, \{2\}, \{1, 2\}$ for $|\mathcal{L}^{(m)}| = 2$. However, this means that every component classifier needs the all training instances many of which are assigned to the null class. This causes two problems: one is the designing cost that is increased in proportion to the total number of training instances, and another is that an enhanced imbalance problem where the instances of null class occupy the majority in each model, bringing a bad influence to classifier design.

TABLE I: Classification process of a test instance in RAkEL.

	Classification result in each label subset					
Label subset	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
$\mathcal{L}^{(1)} = \{\lambda_1, \lambda_2, \lambda_6\}$	1	1	–	–	–	0
$\mathcal{L}^{(2)} = \{\lambda_2, \lambda_4, \lambda_5\}$	–	1	–	0	0	–
$\mathcal{L}^{(3)} = \{\lambda_4, \lambda_5, \lambda_6\}$	–	–	–	0	0	1
$\mathcal{L}^{(4)} = \{\lambda_2, \lambda_3, \lambda_6\}$	–	0	1	–	–	0
$\mathcal{L}^{(5)} = \{\lambda_3, \lambda_4, \lambda_5\}$	–	–	1	0	1	–
Average votes	$1/1$	$2/3$	$2/2$	$0/3$	$1/3$	$1/3$
Final prediction	1	1	1	0	0	0

In this paper, we solve this multiple usage of the whole data by establishing an upper-level classification. We build individual classifiers using the only instances assigned to that model. That is, we do not use instances assigned to the null

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

class. Instead, we select models in accord with a given test instance in each round. The difference is illustrated in Fig. 1. A training instance is distributed into multiple models such that it shares at least one label with the label subset of the model. We carry out such an assignment by regression for a test instance. For example, in Table I, ideally, an instance with hidden labels $\{\lambda_1, \lambda_2, \lambda_3\}$ is distributed to four models except for the third and majority voting is performed on the four models.

Notations The goal of MLC is to learn the relationship between a F-dimensional vector $\boldsymbol{x} \in \mathbb{R}^F$ and a L-dimensional binary vector $\boldsymbol{y} \in \{0,1\}^L$ from N training instances $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)})\}$, and to predict a label vector $\hat{\boldsymbol{y}}$ for a test instance $\boldsymbol{x}$ from the relationship. For simplicity, we use $\mathbf{X} = \left[\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right]^T \in \mathbb{R}^{N \times F}$ as a feature matrix and $\mathbf{Y} = \left[\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\right]^T \in \{0,1\}^{N \times L}$ as a label matrix.

We use $\mathcal{L} = \{\lambda^{(1)}, \ldots, \lambda^{(L)}\}$ as a set of labels, $\mathcal{L}^{(m)}$ as mth sampled label subset of size $|\mathcal{L}^{(m)}|$ and $\mathcal{P}(\mathcal{L}^{(m)})$ as its power-set. We use $\boldsymbol{w}^{(m)} = \left[\boldsymbol{w}_1^{(m)}, \boldsymbol{w}_2^{(m)}, \ldots, \boldsymbol{w}_L^{(m)}\right] \in \{0,1\}^L$ as a membership vector for mth model with $\mathcal{L}^{(m)}$ such that $\boldsymbol{w}_i^{(m)}$ takes 1 if $\lambda_i \in \mathcal{L}^{(m)}$, otherwise 0. By $\mathcal{D}^{(m)} \subseteq \mathcal{D}$, we denote the subset of training instances for mth model with $\mathcal{L}^{(m)}$. We sample randomly M label subsets of a constant size K, $|\mathcal{L}^{(m)}| = K$ $(m = 1, 2, \ldots, M)$.

II. Related Work

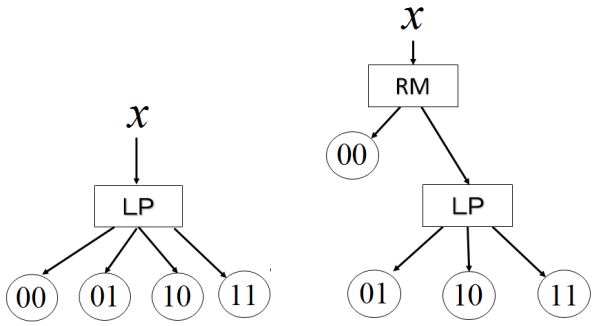
A meta-label, a reasonable combination of labels, is generally accepted idea in MLC field and its typical construction and usage is as follows [7]:

1) (*Partitioning*): Partition a label set $\mathcal{L}$ into M meta-labels (label subsets) $\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(M)}$.
2) (*Relabeling*): Choose reasonable combinations in each $\mathcal{L}^{(m)}$ $(m = 1, 2, \ldots, M)$. Usually a limited number of combinations is adopted, but in LP methods, all possible label subsets are chosen within $\mathcal{L}^{(m)}$.
3) (*Recombination*): Decompose a meta-label into a set of original labels as the candidates of the output.

In the partitioning step, RAkEL repeats M times a random sampling in a fixed size K. This achieves a faster computation compared to the other partitioning methods by clustering [6], chi-square testing [8], conditional dependencies of labels [5] and solving a cover-set problem [9], [10].

In the relabeling step, Read *et al.* proposed a pruning to reduce the number of classes [20]. It prunes the less-frequent classes in the LP method. In [7], pruning on label subsets is proposed in RAkEL. This method enables us to handle even a large size of label subsets even in RAkEL.

In the recombination step, several variants of RAkEL have been proposed so far. Some improve the voting scheme by optimizing the classification accuracy in a cross validation [11]. RAkEL++ uses a confidence value instead of a majority voting [10]. Gharroudi *et al.* choose the value of a threshold of majority voting for each labels greedily [12].



(a) The original RAkEL (b) The proposed fRAkEL

Fig. 1: Difference of classification processes in a model of a label subset of size two. LP denotes a Label-Power model and RM denotes a regression model.

In addition, some other methods try to incorporate random-forest type approaches with RAkEL [7], [13]. They sample instances, features and labels at the same time. At the expense of an improved accuracy, they need more computational time.

In contrast to label reduction approaches above, some methods reduce the number of instances by clustering [14]. Unfortunately, it costs a large amount of computational time too. In this paper, we reduce both labels and instances, where each model consists of a small number of labels such as RAkEL and is learned from a limited number of instances sharing labels with each model.

The label space dimension reduction (LSDR) [15] is also related to our approaches. It can be considered as a soft clustering of labels. However, due to its soft membership manner, the number of labels in each model cannot be reduced. Therefore, instead of LP, BR methods are used for the classification in each model. Several variants of LSDR with traditional dimension reduction methods such as PCA and CCA [16]–[18] have been considered.

III. The Proposed Model

We first provide a brief review of RAkEL [4].

A. A Brief Review of Random k Label Set (RAkEL)

RAkEL is one of meta-label learning methods which allows overlapping between meta-labels. RAkEL carries out the following steps:

1) Sample K labels at random from $\mathcal{L}$ $(K \ll L)$. Repeat this M times to have $\mathcal{L}^{(m)}$ $(m = 1, 2, \ldots, M)$.
2) Construct an LP classifier with $\mathcal{P}(\mathcal{L}^{(m)})$ classes and the null class using all instances in $\mathcal{D}^{(m)} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)} \wedge \boldsymbol{w}^{(m)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)} \wedge \boldsymbol{w}^{(m)})\}$ where many of which are assigned to the null class.
3) Combine the classification results on a test instance by majority voting.

RAkEL is known as one of best LP based methods. However, as we stated, RAkEL is not efficient since it needs all instances for designing each component LP classifier (See Step 2).

Algorithm 1 The proposed fast RAkEL (fRAkEL)

Input: Label matrix $\mathbf{Y}$,Feature matrix $\mathbf{X}$, Size of label subsets K, Number of model M, Base LP classifier $f()$, Ridge parameter α, Threshold parameter β, A test instance $\boldsymbol{x}$;
Output: Predicted label vector $\hat{\boldsymbol{y}}$;
{**Training**}
for $m = 1$ to M **do**
  $\mathcal{L}^{(m)} \leftarrow$ randomsampling(L, K);
  $\boldsymbol{w}^{(m)} \leftarrow (\boldsymbol{w}_1^{(m)}, \boldsymbol{w}_2^{(m)}, \ldots, \boldsymbol{w}_L^{(m)}) \in \{0,1\}^L$ where $\boldsymbol{w}_i^{(m)} = 1$ if $\lambda^{(i)} \in \mathcal{L}^{(m)}$;
  $\mathcal{D}^{(m)} \leftarrow \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)} \wedge \boldsymbol{w}^{(m)}) | \boldsymbol{y}^{(n)T}\boldsymbol{w}^{(m)} \geq 1\}$;
  Construct $f^{(m)}$ with $\mathcal{D}^{(m)}$;
end for
Construct a target matrix $\mathbf{Z}$ according to (1);
Learn $\mathbf{V}$ by minimizing $\min_{\mathbf{V}} \|\mathbf{Z} - \mathbf{X}\mathbf{V}\|_2^2 + \alpha\|\mathbf{V}\|_2^2$;
{**Testing**}
Obtain assignment $\hat{\boldsymbol{z}} = \text{round}(\boldsymbol{x}^T\mathbf{V}, \beta)$;
Apply all classifiers $f^{(m)}$ of $\hat{\boldsymbol{z}}_m = 1$ and combine the output $\hat{\boldsymbol{y}}$ by majority voting;

B. Framework

We solve this problem by two-stage classification. In the training phase, as shown in Fig. 1(b), at the first stage we distribute every instance to a limited number of models only. At the second stage, we learn an LP classifier in each model with the distributed instances.

The detailed steps are as follows:

1) Sample K labels from $\mathcal{L}$ ($K \ll L$). Repeat this M times.
2) Build $\mathcal{D}^{(m)}$ by collecting instances with labels intersecting $\mathcal{L}^{(m)}$ ($m = 1, 2, \ldots, M$).
3) Construct an LP classifier in each model m with $\mathcal{D}^{(m)}$.
4) For a test instance, choose the models in the following way and combine the outputs by majority voting.

Here, Step 2 and Step 4 are the key processes of the proposed method. We simply choose the training instances which share at least one label with $\mathcal{L}^{(m)}$ in Step 2. With the membership vector $\boldsymbol{w}^{(m)}$, $\mathcal{D}^{(m)}$ is defined as

$$\mathcal{D}^{(m)} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)} \wedge \boldsymbol{w}^{(m)}) | \boldsymbol{y}^{(n)T}\boldsymbol{w}^{(m)} \geq 1\}.$$

In the testing phase, for the simplicity and the small computation cost, we use a regression for the model selection in Step 4. First, we construct an instance-model relation matrix $\mathbf{Z} \in \{0,1\}^{N \times M}$ from the training instances defined as

$$\mathbf{Z} = (\mathbf{Z}_{nm}) = \begin{cases} 1 & (\boldsymbol{y}^{(n)T}\boldsymbol{w}^{(m)} \geq 1) \\ 0 & (otherwise). \end{cases} \quad (1)$$

Regarding $\mathbf{Z}$ as the target matrix, we carry out *Ridge Regression* of $\mathbf{Z}$ on $\mathbf{X}$:

$$\begin{gathered} \min_{\mathbf{V}} \|\mathbf{Z} - \mathbf{X}\mathbf{V}\|_2^2 + \alpha\|\mathbf{V}\|_2^2, \quad (2) \\ \mathbf{Z} \in \{0,1\}^{N \times M},\ \mathbf{X} \in \mathbb{R}^{N \times F},\ \mathbf{V} \in \mathbb{R}^{F \times M}, \end{gathered}$$

where α is the ridge parameter. Finally, we predict the membership $\hat{\boldsymbol{z}}$ of $\boldsymbol{x}$ by

$$\hat{\boldsymbol{z}} = \text{round}(\boldsymbol{x}^T\mathbf{V}, \beta),\ \text{where round}(x,\beta) = \begin{cases} 1 & (x \geq \beta) \\ 0 & (\text{otherwise}). \end{cases}$$

Here, β is a parameter.

A pseudo-code of the proposed framework is described in Algorithm 1. Note that the regression part can be replaced to the other regressions including non-linear regression or the other binary classification algorithms. In addition, the proposed framework can be applicable for the other RAkEL based algorithms such as [7], [10], [13].

C. Complexity Analysis

In the training phase, the complexity of RAkEL is $O(MT_1)$ where M is the number of models and T_1 denotes the complexity of the base learner for training. The proposed algorithm has the same complexity $O(MT_1)$, however, the complexity of the base learner T_1 is largely reduced. For example, if we use a linear SVM, with one vs. the rest strategy, the complexity T_1 is $O(FN^2 2^K)$. Since the proposed algorithm reduces the number of training instances from N to $N^{(m)} = |\mathcal{D}^{(m)}|$, T_1 is α^2 times less than that of RAkEL for $\alpha = N^{(m)}/N$. If any pair of instances does not share labels in the label subsets, the expected number of instances used in each model is calculated as

$$\mathbb{E}(N^{(m)}) = BNK, \quad (3)$$

where B is the label-density defined as

$$B = \frac{1}{NL}\sum_{i,j}\mathbf{Y}_{ij}.$$

For example, on the *corel5k* dataset, since the label-density is about $B = 0.03$ (see Table II), the expected number of instances used in each model is $0.03NK$. Therefore, the proposed algorithm with a linear SVM and $K = 3$ is about $(1/\alpha)^2 = (N/\mathbb{E}(N^{(m)}))^2 = (1/(BK))^2 = (1/0.09)^2 \simeq 123.4$ times faster than the original RAkEL theoretically, although the actual ratio was about 30 times.

In the testing phase, the complexity of RAkEL is $O(MT_2)$ where T_2 denotes the complexity of the base learner for testing. On the other hand, the proposed algorithm needs $O(F + \|\hat{\boldsymbol{z}}\|_1 T_2)$, assuming the regression matrix is already learned (Step 10 in Algorithm 1). Since the complexity T_2 is larger than $O(F)$ in general, if the number of selected models, $\|\hat{\boldsymbol{z}}\|_1$, is less than m, the proposed algorithm is faster than RAkEL.

As shown above, the proposed algorithm is more advantageous if we use classifiers with a larger complexity in N. For example, if we use a RBF-kernel SVM, $T_1 = O(FN^2)$ and $T_2 = O(FN)$, thus, fRAkEL is 123.4 times faster in the training phase than RAkEL and more than 11.1 times faster in testing phase on the *corel5k*.

TABLE II: Dataset used in the experiment.

Dataset	N	L	F	B	C	LD
scene	2407	6	294	0.18	1.07	15
emotions	593	6	72	0.31	1.87	27
genbase	662	27	1186	0.05	1.25	32
yeast	2417	14	103	0.30	4.23	198
medical	978	45	1449	0.03	1.25	94
enron	1702	53	1001	0.06	3.38	753
CAL500	502	174	68	0.15	26.04	502
corel5k	5000	374	499	0.01	3.52	3175
bibtex	7395	159	1836	0.02	2.40	2856
tmc2007	28596	22	500	0.10	2.16	1341
mediamill	43907	101	120	0.04	4.38	6555
delicious	16105	983	500	0.02	19.02	15806

IV. Experiments

A. Dataset and Evaluation Measurement

We conducted experiments on twelve benchmark datasets which are summarized in Table II [19].[1] In Table II, B, C and LD denote the label-density, label-cardinality and the number of distinct label subsets, respectively.

To evaluate the results, we used Macro-F1 and Micro-F1 as the evaluation measurements defined as

$$\textbf{Macro F1}: = \frac{1}{L}\sum_{j=1}^{L}\frac{2\sum_{i=1}^{N}\hat{\mathbf{Y}}_{ij}\cdot\mathbf{Y}_{ij}}{\sum_{i=1}^{N}\hat{\mathbf{Y}}_{ij}+\sum_{i=1}^{N}\mathbf{Y}_{ij}},$$

$$\textbf{Micro F1}: = \frac{2\sum_{j=1}^{L}\sum_{i=1}^{N}\hat{\mathbf{Y}}_{ij}\cdot\mathbf{Y}_{ij}}{\sum_{j=1}^{L}\sum_{i=1}^{N}\hat{\mathbf{Y}}_{ij}+\sum_{j=1}^{L}\sum_{i=1}^{N}\mathbf{Y}_{ij}},$$

where, N is the number of test instances, L is the number of labels, $\hat{\mathbf{Y}}$ is the predicted label matrix and $\mathbf{Y}$ is the ground-truth label matrix. By taking an average over labels, Macro-F1 takes into account the less-frequent labels evenly, while Macro-F1 does not. We repeated five trials with different random label subsets and calculated the averaged values of the measurements.

B. Comparing Methods and Parameters

We compared the following three methods including our proposal:[2]

1) LP: Label-Power method
2) RAkEL: Random K label subsets [4]
3) fRAkEL: Proposed method

We used a linear SVM implemented in liblinear [21] as a base learner in all algorithms because of its fast computation. For RAkEL and the proposed fRAkEL, we employed the same setting in [4], [8], [13], and used the size of label subsets $K = 3$ and the number of label subsets $M = 2L$. Note that LP is a special case of RAkEL with $K = L$, $M = 1$. We implemented all algorithms on MATLAB.[3] For the proposed fRAkEL, we used ridge parameter $\alpha = 0.1$ for all datasets and β was tuned by five cross validation on the training dataset.

C. Results

First, we see in Table III that the averaged number of instances used in each model is reduced to $3\% - 74\%$ of the total number of instances. Similarly, the averaged number applied classifiers for each test instance is also reduced to $5\% - 81\%$ of the number of all classifiers. Especially, on the large datasets ($N \geq 5000$) as estimated as BK by (3), the average number of $N^{(m)}$ were less than 27% of N.

Table IV shows the execution time on all datasets. The proposed fRAkEL was faster than the original RAkEL on all datasets. However, the proposed fRAkEL was slower than the LP method on five datasets. This is because LP uses distinct label subsets as classes in practice, thus, in practice the complexity depends on the number of LD. On these five datasets, LD is far smaller than the possible combination of 2^L. In such cases, the cost of constructing $M = 2L$ models becomes larger than LP methods (*e.g.* fRAkEL and RAkEL needed to construct 348 models and each model has less than 2^3 classes on *CAL500* dataset). Otherwise, LP is the slowest. As explained above, when the label number L is large and the label density B is low such as *corel5k* ($L = 374, B = 0.03$), *bibtex* ($L = 159, B = 0.02$) and *delicious* ($L = 983, B = 0.05$), the proposed fRAkEL is 5 to 30 times faster than RAkEL.

Table V and Table VI show the accuracy by Macro-F1 and Micro-F1. The proposed fRAkEL and RAkEL outperformed LP on all datasets except *yeast*. This is consistent to the result of previous reports [4], [7]. This is probably because ensemble did not work for this particular problem. The proposed fRAkEL outperformed RAkEL on all datasets except for *enron* and *medical* and in Micro-F1. Even in *medical* and *enron* datasets, the difference is slight.

This shows that the proposed fRAkEL succeeded to avoid the imbalance problem that RAkEL possessed. However, it turned out that fRAkEL has another problem in the regression step as shown below. We compared the values of Accuracy of component LP classifiers on RAkEL (with the null class) and fRAkEL (without the null class) defined as

$$\textbf{Accuracy:} = \frac{TP+TN}{TP+TN+FP+FN},$$

where TP, TN, FP, and FN denotes the number of True-Positive, True-Negative, False-Positive and False-Negative examples, respectively. For investigation of the goodness of training instance distribution (Step 2), we assign the test instances to the correct models, sharing the labels, instead of assigning by regression. We picked one small size dataset (*scene*), two medium size datasets (*yeast, enron*) and two large size datasets (*corel5k, bibtex*) for this comparison. Table VII shows the averaged values of Accuracy of component LP classifiers on these five datasets. On the datasets which have a large null class (*e.g. enron*, *corel5k* and *bibtex* dataset, see Table III), the classification accuracy of component LP

[1] All datasets are available at http://mulan.sourceforge.net/datasets-mlc.html

[2] The other acceleration methods or more accurate methods such as [7], [9], [10], [13], [20] were not compared for simplicity.

[3] https://github.com/KKimura360/fast_RAkEL_matlab

TABLE III: The average number of training instances used in a classifier and the average number of applied classifiers for a test instance.

Sample ratio	Dataset											
	scene	emotions	genbase	yeast	medical	enron	CAL500	corel5k	bibtex	tmc2007	mediamill	delicious
$\lvert\mathcal{D}^{(m)}\rvert/\lvert\mathcal{D}\rvert$	.51±.02	.73±.09	.12±.09	.69±.17	.08±.08	.17±.17	.37±.22	.03±.04	.04±.02	.26±.21	.12±.18	.05±.06
$\lVert\hat{\boldsymbol{z}}\rVert_1/m$	.62±.01	.81±.13	.13±.01	.82±.04	.12±.01	.22±.02	.57±.01	.04±.00	.03±.00	.29±.03	.12±.01	.07±.00

TABLE IV: Execution time (seconds)

Algorithm	Dataset											
	scene	emotions	genbase	yeast	medical	enron	CAL500	corel5k	bibtex	tmc2007	mediamill	delicious
LP	**0.69**	**0.06**	**0.07**	7.69	**0.22**	3.06	**0.75**	212.07	40.79	110.18	NA (>1000)	NA (>1000)
RAkEL	4.30	0.21	0.27	5.26	0.80	7.95	4.17	61.06	36.47	23.48	468.03	900.86
Proposal	2.00	0.18	0.11	**2.82**	0.26	**1.33**	2.05	**2.32**	**3.75**	**6.95**	**54.74**	**62.38**

TABLE V: Macro-F1

Algorithm	Dataset											
	scene	emotions	genbase	yeast	medical	enron	CAL500	corel5k	bibtex	tmc2007	mediamill	delicious
LP	.67±.03	.59±.03	.69±.04	.40±.01	.37±.03	.17±.01	.15±.00	.03±.00	.24±.00	**.61±.00**	NA	NA
RAkEL	.68±.02	.63±.02	.72±.03	**.47±.00**	.39±.02	.19±.01	.11±.01	.04±.00	**.33±.00**	.57±.00	.05±.00	.09±.00
Proposal	**.70±.02**	**.65±.02**	**.73±.02**	**.47±.01**	**.40±.02**	**.20±.01**	**.21±.00**	**.05±.00**	**.33±.00**	**.61±.00**	**.11±.00**	**.13±.00**

TABLE VI: Micro-F1

Algorithm	Dataset											
	scene	emotions	genbase	yeast	medical	enron	CAL500	corel5k	bibtex	tmc2007	mediamill	delicious
LP	.66±.03	.60±.02	.97±.00	**.62±.00**	.75±.02	.46±.01	.33±.01	.14±.00	.32±.00	.68±.00	NA	NA
RAkEL	.67±.02	.65±.01	.98±.00	.59±.00	**.78±.00**	**.49±.01**	.40±.01	.14±.00	.41±.00	.70±.00	.54±.00	.23±.00
Proposal	**.69±.03**	**.66±.02**	**.99±.00**	.61±.01	.77±.01	.48±.02	**.46±.00**	**.27±.00**	**.45±.00**	**.71±.00**	**.58±.00**	**.37±.00**

classifiers of fRAkEL is far better than that of fRAkEL. This result implies that our model selection and training with the assigned training instances work well. Next we investigated the correctness of the ridge regression by Threat-Score defined as

$$\textbf{Threat-Score:} = \frac{TP}{TP + FP + FN}.$$

Table VIII shows that Threat-Score on five datasets. The values of Threat-Score are low, especially on large datasets. These two facts mean that the performance of fRAkEL is limited by the regression step not by the performance of component LP classifiers.

In summary, the proposed fRAkEL could avoid the imbalance problem that RAkEL possess in each component LP classifier and performs faster than RAkEL with higher classification accuracy. However, fRAkEL has another problem in the regression. This problem would be lesser by employing the other more suitable regression methods such as Logistic regression with sacrificing computational time instead of the ridge regression.

We conducted a sensitivity analysis on the parameter K, the size of label subset, and M, the number of models on *enron* dataset. Fig. 2 and Fig. 3 show the results. On the previous experiments with $K = 3$ and $M = 2L$, the proposed algorithm performed worse w.r.t. both Micro-F1. However, we see that the proposed fRAkEL can perform better by choosing a value of K larger than 4. Fig.4 shows the execution time with different values of K and M. The proposed fRAkEL is more sensitive than RAkEL. This is consistent to our analysis. Nevertheless, fRAkEL is still advantageous to RAkEL in large K and large M w.r.t. computational time.

TABLE VII: The averaged Accuracy of component LP classifiers.

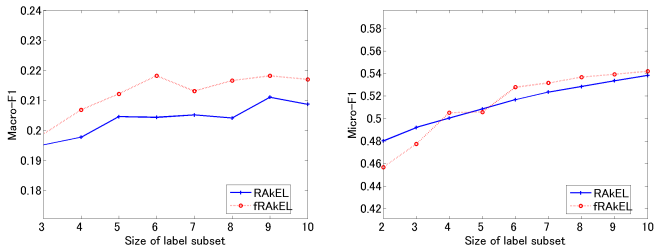
LP	Dataset				
	scene	yeast	enron	corel5k	bibtex
RAkEL	.63±.07	.41±.46	.29±.20	.10±.12	.27±.17
Proposal	**.79±.92**	**.46±.05**	**.73±.16**	**.73±.20**	**.87±.08**

TABLE VIII: The averaged correctness of regression.

RM	Dataset				
	scene	yeast	enron	corel5k	bibtex
Proposal	.67±.01	.70±.02	.32±.02	.16±.00	.28±.01

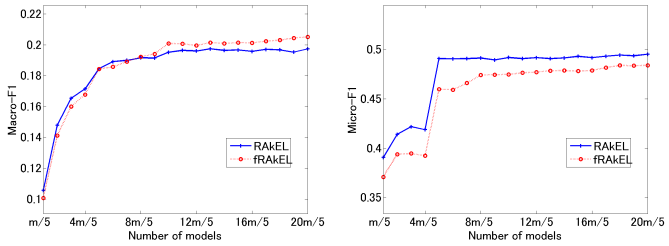
V. Conclusion

In this paper, we have proposed a fast algorithm for Random k-labelsets strategy for multi-label classification. The proposed algorithm employs a two-stage classification to reduce the number of instances used in each local model in addition of



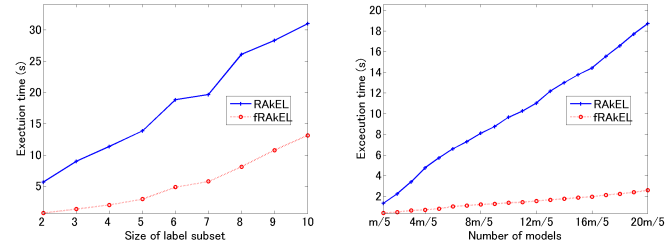
(a) Macro-F1 (b) Micro-F1

Fig. 2: The result with different size K of label subsets on *enron* dataset ($M = 2L$).



(a) Macro-F1 (b) Micro-F1

Fig. 3: The result with different number M of models on *enron* dataset ($K = 3$).



(a) Different size K of models (b) Different number M of models

Fig. 4: The result of execution time with different settings.

a small number of labels. In the experiments on large-scale datasets, the proposed algorithm worked about 30 times faster at the best case than the original RAkEL algorithm, and even improved the accuracy on most of all datasets.

ACKNOWLEDGMENT

This work was partially supported by JSPS KAKENHI Grant Number 14J01495 and 15H02719.

REFERENCES

[1] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[2] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*. Springer, 2010, pp. 667–685.

[3] 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, 2014.

[4] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[5] J. Read, C. Bielza, and P. Larrañaga, "Multi-dimensional classification with super-classes," *IEEE Transactions on Knowledge and Data Engineering,*, vol. 26, no. 7, pp. 1720–1733, 2014.

[6] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008, pp. 30–44.

[7] J. Read, A. Puurula, and A. Bifet, "Multi-label classification with meta-labels," in *IEEE International Conference on Data Mining*. IEEE, 2014, pp. 941–946.

[8] L. Tenenboim-Chekina, L. Rokach, and B. Shapira, "Identification of label dependencies for multi-label classification," in *Working Notes of the Second International Workshop on Learning from Multi-Label Data*, 2010, pp. 53–60.

[9] L. Rokach and E. Itach, "An ensemble method for multi-label classification using an approximation algorithm for the set covering problem," in *Proceedings of the 2nd International Workshop on Learning from Multilabel Data (MLD)*, 2010, pp. 37–44.

[10] L. Rokach, A. Schclar, and E. Itach, "Ensemble methods for multi-label classification," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7507–7523, 2014.

[11] R.-E. Fan and C.-J. Lin, "A study on threshold selection for multi-label classification," *Department of Computer Science, National Taiwan University*, pp. 1–23, 2007.

[12] O. Gharroudi, H. Elghazel, and A. Aussem, "Calibrated k-labelsets for ensemble multi-label classification," in *Neural Information Processing*. Springer, 2015, pp. 573–582.

[13] G. Nasierding, A. Z. Kouzani, and G. Tsoumakas, "A triple-random ensemble classification method for mining multi-label data," in *IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2010, pp. 49–56.

[14] G. Nasierding, G. Tsoumakas, and A. Z. Kouzani, "Clustering based multi-label classification for image annotation and retrieval," in *IEEE International Conference on Systems, Man and Cybernetics, 2009*. IEEE, 2009, pp. 4514–4519.

[15] D. Hsu, S. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing." in *Advances in Neural Information Processing Systems 22*, 2009, pp. 772–780.

[16] Y.-N. Chen and H.-T. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 1529–1537.

[17] F. Tai and H.-T. Lin, "Multilabel classification with principal label space transformation," *Neural Computation*, vol. 24, no. 9, pp. 2508–2542, 2012.

[18] Y. Zhang and J. G. Schneider, "Multi-label output codes using canonical correlation analysis," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 873–882.

[19] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *The Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.

[20] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *IEEE International Conference on Data Mining*. IEEE, 2008, pp. 995–1000.

[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.