2013 IEEE 13th International Conference on Data Mining

Leveraging Supervised Label Dependency Propagation for Multi-label Learning

Bin Fu*†, Guandong Xu†, Zhihai Wang*, and Longbing Cao†
*School of Computer and Information Technology
Beijing Jiaotong University, Beijing, 100044, China
Email: {09112072, zhhwang}@bjtu.edu.cn
†Advanced Analytics Institute
University Technology Sydney
Email: {Guandong.Xu, LongBing.Cao}@uts.edu.au

***Abstract*—Exploiting label dependency is a key challenge in multi-label learning, and current methods solve this problem mainly by training models on the combination of related labels and original features. However, label dependency cannot be exploited dynamically and mutually in this way. Therefore, we propose a novel paradigm of leveraging label dependency in an iterative way. Specifically, each label's prediction will be updated and also propagated to other labels via an random walk with restart process. Meanwhile, the label propagation is implemented as a supervised learning procedure via optimizing a loss function, thus more appropriate label dependency can be learned. Extensive experiments are conducted, and the results demonstrate that our method can achieve considerable improvements in terms of several evaluation metrics.**

***Keywords*-Multi-label learning; Label dependency; Random walk with restart**

I. Introduction

In real applications, an instance often has multiple labels simultaneously. For instance, a film can be tagged as *thriller*, *action* and crime; a book can be categorized as *science* and *education* etc. Accordingly, learning to predict labels for such kind of instance is called multi-label learning. Nowadays, multi-label learning is already playing a crucial role in various applications, e.g., text categorization [1], scene classification [2], and recommender systems [3], etc.

Currently, exploiting dependency between labels has been recognized as an effective strategy to boost the learning performance [4]. Generally, label dependency refers to each label's prediction is also determined by other labels apart from the instance's features. A toy example is given in Fig.1, where $\{X_i\}_{1 \ll i \ll d}$ denote the features of instances, and $\{Y_1, Y_2, \ldots, Y_5\}$ is a set of labels. It shows that each label's value also depends on some other labels apart from the features. Various approaches have been proposed with the motivation of capturing the potential dependencies between labels [5]–[9]. Most of them essentially adopt the similar paradigm to train a predictive model for each label, i.e., $P_X(Y_i) = f_i(X_1, X_2, \ldots, X_d, dep(Y_i))$, here $P_X(Y_i)$, ranges between 0 and 1, indicates Y_i's probability of being instance X's true label, and $dep(Y_i)$ is the set of labels on which Y_i depends. Thus, the multi-label problem is converted to multiple single-label problems, and label dependencies are exploited via adding dependent labels into the feature set. Despite the improvements of these methods, there still are some potential issues need further concern, which include: (1) the *skewness* problem. In practice, the features' number is usually huge, whereas every label might depends on a few labels. For example, label Y_1 in Fig. 1 depends only on Y_2, whereas the number of features d could be thousands or even more. Thus, the learning process will be appreciably biased towards the features when integrating dependent labels with features, and the impact of label dependency would be depressed greatly; (2) Label dependency should not only be direct (e..g, label Y_2 and Y_3 in Fig.1), but also indirect (as seen in Fig.1 label Y_1 and Y_3 should be related through label Y_2), and each label's prediction should also evolve with the update of other dependent labels. Hence the labels' predictions should be updated in a process similar to the Markov chain that iterates until reaches the stable status. (3) The label dependencies are not exploited towards optimizing an explicit loss function in most methods.

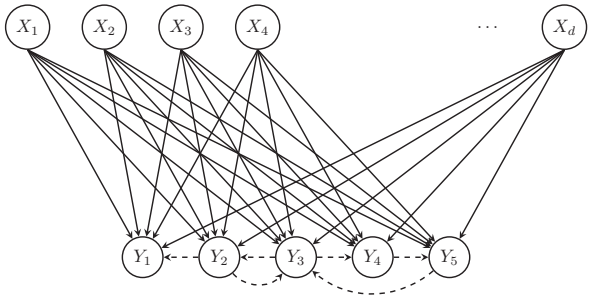

Figure 1. A possible relationship between features and labels

To solve above issues, we thereby propose a new optimization framework in this paper called *SELD* (Supervised Exploiting Label Dependency) for multi-label learning. To summarize, our main contributions are as follows.

(1) We design a novel paradigm to intensify the impact of label dependency, in which $dep(Y_i)$, are exploited separately through the paradigm shown in Eqn.(1).

$$P(Y_i) = \alpha \cdot h_i(dep(Y_i)) + (1 - \alpha) \cdot f_i(X_1, \ldots, X_d) \quad (1)$$

Corresponding author: Guandong Xu

1550-4786/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDM.2013.143

Here h_i is function which determines the impact of $dep(Y_i)$ on label Y_i, and f_i is a predictive model that trained only on the features. Label dependency thus will be exploited properly by learning an appropriate h_i and the parameter α.

(2) We apply the RWR (Random Walk with Restart) strategy to implement the process of label updating and propagation. All labels' predictions will be updated simultaneously by predictions propagated from other labels until convergence, thus a globally good result might be achieved.

(3) A function is designed to quantify the dependencies between labels based on a set of extracted features. The benefit is that the optimal parameters of this function can be learned while guiding the random walk process to minimize a particular loss function explicitly. To the best of our knowledge, this is the first work to exploit label dependency in a supervised way.

(4) We also conduct empirical investigations on diverse data sets, and compare our approach with other state-of-the-art methods using different metrics.

The remainder of paper is organized as follows. Section 2 reviews related work. The definition of multi-label learning and the objective of our approach are introduced in Section 3. We detail our proposed approach in Section 4. Experimental design and results are presented in Section 5, followed by the conclusions in Section 6.

II. Related work

There are various approaches that exploit different types of label dependency [4], categorized as follows.

(1) Training a predictive model for each label on combination of common features and dependent labels. This is the most commonly-used strategy, and the key step is for each label to determine the labels on which it depends. Some methods assume that every label depends on all other labels [5], while others determined the label dependencies explicitly using various models. For example, both *classifier chain* and *probabilistic classifier chain* ranked the labels randomly and assumed each label depended on all its preceding labels [6], [7]. Due to the randomness, some weak dependencies might be included while strong dependencies are ignored. Zhang et al. presented the LEAD method that adopted a specific Bayesian network to encode the conditional dependencies of the labels [8]. Other models used to describe label dependencies include the multi-dimensional Bayesian network, and topic model etc. [10], [11]. For every label, these methods can explicitly determine the labels on which it depends.

(2) Exploiting label dependency in the phase of predicting labels for instances. Instead of exploiting label dependency in the training phase as above, this paradigm exploits label dependency to constrain the labels' predictions directly. Park et al. adopted association rules to discover label constraints from data and use them to correct initial predictions [12]. Guo et al. proposed the CDN-LR method which used Gibbs sampling to obtain the final predictions after being given the initial predictions [9]. Our proposed method also complies with this paradigm, but the difference that is we use the RWR strategy to update the labels' predictions. In particular, we adapt the supervised RWR strategy [13] to exploit label dependency towards optimizing a specific loss function, which has not been fully studied yet to our knowledge.

III. Problem Statement

A. Multi-Label Learning

Multi-label learning can be formally defined as follows. Let X denote the instance space and $Y = \{y_1, y_2, \ldots, y_m\}$ be a set of labels, thus a training set can be represented by $D = \{(x_i, C_i)|1 \leq i \leq n\}$ where $x_i \in X$ is an instance and $C_i \subseteq Y$ is a subset of labels associated with x_i. The objective of multi-label learning is to train a classifier: $f : X \rightarrow 2^C$ based on D, which is then used to predict labels for new instances. C_k can also be represented by a Boolean label vector $< y_{k1}, y_{k2}, \ldots, y_{km} >$, where $y_{kj} = 1$ indicates label y_j is x_k's relevant label ($y_j \in C_k$), otherwise $y_{kj} = 0$.

A multi-label problem can be transformed to $|m|$ single-label problems, one corresponds to a label. We thereby can train a classifier f_i for each label y_i. In most cases, f_i will output a value between 0 and 1 that predicts y_i's probability of being an instance's true label. Our approach will also adopt this framework.

B. Our Targeted Problem

Arguably, the dependency degree between labels should be determined through minimizing a specific loss function. We thus define the overall loss function of a classifier f as:

$$E(f, D) = \sum_{i=1}^{|D|} E_i \tag{2}$$

Here E, the overall loss over D, is decomposed as the sum of the loss on every single instance, i.e., E_i. Given a vector $y_i =< y_1(x_i), y_2(x_i), \ldots, y_m(x_i) >$, where $y_k(x_i)$'s value is 1 or 0 to indicate whether label y_k is x_i is a true label or not, and a vector $p_i =< p(y_1|x_i), p(y_2|x_i), \ldots, p(y_m|x_i) >$, where $p_(y_k|x_i)$ is y_k's probability of being x_i's true label predicted by f, E_i thus can be defined as:

$$E_i = \frac{\sum_{j=1}^{m}(y_j(x_i) - p(y_j|x_i))^2}{2m} \tag{3}$$

The more accurate the probabilities given by a classifier f, the less the loss is. Therefore, the task of our approach is to train a classifier f_i for label y_i towards minimizing Eqn.(2), through exploiting appropriate label dependency.

IV. Our proposed *SELD* approach

A. General Framework

In this section, a novel framework named *SELD* is proposed to exploit label dependency. It consists of two steps

when predicting labels for a test instance x: (1) generating an initial prediction for every label y_i using corresponding predictive models f_i that trained only on features; and (2) updating all these initial predictions simultaneously in an iterative process of label propagation. Specifically, the final probabilistic classifier p_i for label $y_i$is defined as follows:

$$p_i(x) = \alpha \cdot \frac{\sum_{j=1}^{m} p_j(x) \cdot c_{ij}}{m} + (1-\alpha) \cdot f_i(x) \quad (4)$$

Here the output of $p_i(x)$ and $f_i(x)$ is label y_i's final and initial probability of being x's true label respectively, and c_{ij}, in $[0, 1]$, indicates the degree of label y_i's dependency on label y_j, or the probability of label y_j's value propagates to y_i. Thereby, y_i's final prediction is determined by a linear combination of two components, i.e., the final predictions of other labels and the common features. α is the parameter used to balance the influence of these two components.

We then use $p(y_i|x)$ to replace $p_i(x)$. Taking all the labels into consideration, the final probability vector $p(Y|x) = (p(y_1|x), p(y_2|x), \ldots, p(y_m|x))^T$ can be written as Eqn.(5).

$$\begin{pmatrix} p(y_1|x) \\ p(y_2|x) \\ \vdots \\ p(y_m|x) \end{pmatrix} = \frac{\alpha}{m} C \begin{pmatrix} p(y_1|x) \\ p(y_2|x) \\ \vdots \\ p(y_m|x) \end{pmatrix} + (1-\alpha) \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{pmatrix} \quad (5)$$

C is a $m \times m$ matrix, in which c_{ij} is defined as above. To compute $p(Y|x)$, We apply the random walk with restart method to Eqn.(5). It will compute $p(Y|x)$ iteratively in the following way.

$$\begin{cases} p(Y|x)(0) = \frac{1}{m} \cdot \mathbf{1}_{|m|} \\ p(Y|x)(r+1) = \frac{\alpha}{m} \cdot C \cdot p(Y|x)(r) + (1-\alpha) \cdot f(Y|x) \end{cases} \quad (6)$$

Here $p(Y|x)(r)$ is $p(Y|x)$'s value in the rth iteration, and $f(Y|x)$ is the vector consists of every label's initial probability. This process will continue until $p(Y|x)$ converges.

It is clear that several of the most distinct advantages of our approach are: (1) the exploiting of label dependency can be intensified significantly by assigning α an appropriate value; (2) all labels' predictions are updated simultaneously and iteratively, thus an overall good prediction might be achieved after convergence.

B. Optimization Task of Determining Label Dependency

Given above framework, we still need to find the optimal matrix C to minimize the loss function shown in Eqn.(2). To this end, a set of features are extracted to characterize the dependency between labels. Specifically, label y_i's dependency on label y_j is depicted by a d-length feature vector A_{ij}, and the dependency value is computed by function g:

$$c_{ij} = g(y_j \to y_i) = g(\sum_{t=1}^{d} \theta_t \cdot A_{ijt}) = g(\theta^T A_{ij}). \quad (7)$$

Algorithm 1: Process of finding the optimal parameters

Data: Training dataset: $D = \{(x_i, C_i)\}_{1 \le i \le n}$, and the learning rate : lr.
Result: Optimal parameter $\theta^T =< \theta_1, \ldots, \theta_m >$.

```
1 begin
2     Initialize θ(0)
3     r = 0
4     while E(θ) has not converged do
5         shuffle the instances in D randomly
6         for i = 1 to n do
7             for t = 1 to d do
8                 θ_t(r+1) = θ_k(r) − lr · ∂E_i(θ(r))/∂θ_t
9             update matrix C according to Eqn.(7)
```

Here A_{ijt} is the tth element of vector A_{ij}, and θ^T is the parameter that controls the features' weights, thus matrix C is converted to $\{g(\theta^T A_{ij})\}_{m \times m}$. The task now is converted to learn the optimal θ^T to minimize the loss function, which can be formulated as:

$$\min_{\theta} E(\theta) = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} (p_j(x_i) - y_j(x_i))^2}{2m} \quad (8)$$

Alg. 1 details the stochastic gradient descent process that used to solve above task, and line 8 shows the rule of updating θ. The problem now is how to compute $E_i(\theta(r))/\partial\theta_t$. Here $E_i(\theta(r))$ denotes E_i computed using $\theta(r)$. According to Eqn.(3), $E_i(\theta)/\partial\theta_t$ can be written as:

$$\frac{\partial E_i(\theta)}{\partial \theta_t} = \frac{1}{m} \sum_{j=1}^{m} ((p_j(x_i) - y_j(x_i))) \frac{\partial p_j(x_i)}{\partial \theta_t} \quad (9)$$

According to Eqn.(4), $\partial p_i(x)/\partial\theta_t$ can be expressed as:

$$\frac{\partial p_j(x_i)}{\partial \theta_t} = \frac{\alpha}{m} (\sum_{k=1}^{m} c_{jk} \cdot \frac{\partial p_k(x_i)}{\partial \theta_t} + \sum_{k=1}^{m} p_k(x_i) \cdot \frac{\partial c_{jk}}{\partial \theta_t}) \quad (10)$$

Computing $\partial p_i(x)/\partial\theta_t$ becomes the final problem. Since it is difficult to get a close-form solution, we also employ an iterative process to solve it. Combining the partial derivatives of all the labels' predictions together, the partial derivative vector $\partial P(x)/\partial\theta_t = (\frac{\partial p_1(x)}{\partial \theta_t}, \frac{\partial p_2(x)}{\partial \theta_t}, \ldots, \frac{\partial p_m(x)}{\partial \theta_t})^T$ can be formulated as follows.

$$\begin{pmatrix} \frac{\partial p_1(x)}{\partial \theta_t} \\ \frac{\partial p_2(x)}{\partial \theta_t} \\ \vdots \\ \frac{\partial p_m(x)}{\partial \theta_t} \end{pmatrix} = \frac{\alpha}{m} C \begin{pmatrix} \frac{\partial p_1(x)}{\partial \theta_t} \\ \frac{\partial p_2(x)}{\partial \theta_t} \\ \vdots \\ \frac{\partial p_m(x)}{\partial \theta_t} \end{pmatrix} + \frac{\alpha}{m} \frac{\partial C}{\partial \theta_t} \begin{pmatrix} p_1(x) \\ p_2(x) \\ \vdots \\ p_m(x) \end{pmatrix} \quad (11)$$

Note that the computation of $\partial C/\partial\theta_t$ also gets a matrix defined as $\{\partial c_{ij}/\partial\theta_t\}_m \times m$, and c_{ij} in C is determined by

the function $g(\theta^T A_{ij})$ as shown above. Here we choose the sigmoid function for g, whose output will be in the [0, 1] which is suitable for representing the propagation possibility between labels. The definition of g is defined as Eqn.(12).

$$g(\theta^T A_{ij}) = \frac{1}{1 + e^{-\theta^T A_{ij}}} \tag{12}$$

The partial derivative $\frac{\partial c_{ij}}{\partial \theta_t}$ then can be computed as:

$$\frac{\partial c_{ij}}{\partial \theta_t} = \frac{\partial g(\theta^T A_{ij})}{\partial \theta_t} = \frac{e^{-\theta^T A_{ij}}}{(1 + e^{-\theta^T A_{ij}})^2} A_{ijt} \tag{13}$$

Similarly to Eqn.(6), we can adopt an iterative process shown below to compute the final $\partial P(x)/\partial \theta_t$ based on Eqn.(11).

$$\begin{cases} \frac{\partial P(x)}{\partial \theta_t}(0) = \mathbf{1}_{|m|} \\ \frac{\partial P(x)}{\partial \theta_t}(r+1) = \frac{a}{m} C \cdot \frac{\partial P(x)}{\partial \theta_t}(r) + \frac{a}{m} \frac{\partial C}{\partial \theta_t} \cdot P(x) \end{cases} \tag{14}$$

Here $\partial P(x)/\partial \theta_t(r)$ is its value in the rth iteration. The final result is reached when above process converges.

So far, we have described the overall framework of our approach and how to compute the necessary components. It is noted that a theoretical proof of the random walk process' convergence is not emphasized in this paper, but indeed validated in the following experiments.

V. Experiments

A. Data sets and Evaluation Metrics

We conduct experiments on six typical multi-label data sets that described in Table I. For a dataset D, we use $|D|$, $F(D)$, $L(D)$, and $LD(D)$ to denote its number of instances, number of features, number of possible labels, and label cardinality. These data sets come from diverse domains including text, image, biology, and music etc.

Four metrics are used in order to compare these algorithms from different aspects. These metrics are: *Hamming loss*, *one-error*, *coverage*, and *ranking loss*, and their definitions can be found in [14]. For all these metrics, smaller value means a better performance, but they evaluate a method from different aspects. Hamming loss focuses on whether every label is predicted correctly, without taking the relative prediction confidence into consideration. In contrast, the other three metrics evaluate a method based on the predicted ranking of labels for an instance. Since our approach is based on supervised RWR, it will generate a reasonable ranking of labels through giving higher probabilities to true labels. Thus we expect our approach will perform well in these three metrics.

B. Baselines and Parameter Settings

We compare our method with three other methods which exploit label dependencies using different models. They are: IBLR-ML+ [5], Classifier chain (CC) [6]. and CDN-LR [9]. The first two methods employ traditional paradigm that combine the features and dependent labels to train models, while the last one is similar to our approach, but uses Gibbs sampling to exploit the label dependencies.

Table I
Datasets Used in Experiments

name	domain	$\|D\|$	$F(D)$	$L(D)$	$LD(D)$
cal500	music	502	68	174	26.044
emotions	music	593	72	6	1.869
enron	text	1702	1001	53	3.378
rcv1	text	6000	47236	101	2.880
scene	image	2407	294	6	1.074
yeast	biology	2417	103	14	4.237

For IBLR-ML+, we set the number of nearest neighbors to be 10. For CDN-LR, the Gibbs sampling is repeated 100 times, and the next 500 iterations are used to collect samples. In our SELD approach, two features are used to depict label y_i' dependency on label y_j, i.e., (1) the ratio between the number of instances with which both of y_i and y_j are associated and the number of instances with which y_i is associated; and (2) the ratio between the number of instances with which both of y_i and y_j are not associated and the number of instances with which y_i is not associated. *Binary relevance* model [14] is used to compute the initial probability for every label to ensure that label dependencies are only exploited in the RWR process. A threshold is set to 0.5, i.e., only if a label's prediction is greater than 0.5, it is regarded as an relevant label. Logistic regression is used in all four approaches as the base classifier.

C. Results and Analysis

All the algorithms are run on the data sets using five-fold cross validation, and the results reported in this section are the average values.

To begin with, we investigate how exploiting label dependency in different degrees influences the learning on various data sets and metrics. We thus alter α' value from 0.1 to 0.8, and the greater the α, the more weight is given to label dependency. The specific results are shown in Fig. 2-5. It is seen that for ranking-based metrics i.e., one-error, coverage and ranking loss, the α's impact is not significant on the learning performance. The reason is ranking-based metrics are not very sensitive to the absolute value of every label's probability, but only rely on the ranking of them. However, it is clear that exploiting label dependency in different degrees will obviously influence Hamming loss. This is because the computation of Hamming loss relies on the absolute values of every label's prediction, and the varying of α indeed change the value of every label's probability. Specifically, when α becomes larger which means the importance of label dependency is intensified, the performance of our approach is improved on dataset *cal500*, *enron*, and *rcv1*, but worse on the other three datasets. We check Table I, and find that these three data sets have a larger number of labels. This means that there might be complex dependencies between

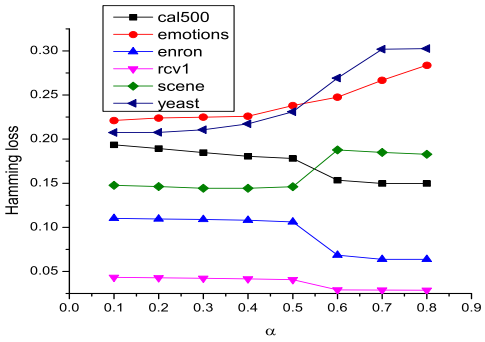

Figure 2. The effect of α in terms of Hamming loss

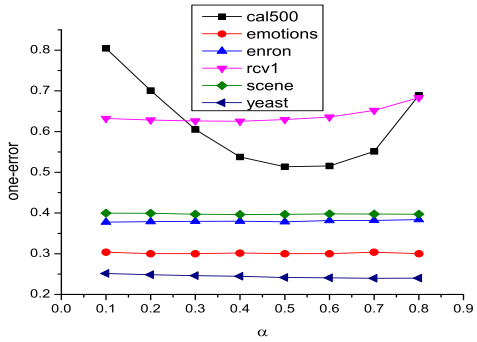

Figure 3. The effect of α in terms of one-error

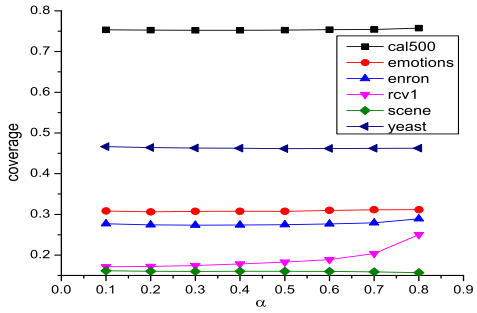

Figure 4. The effect of α in terms of coverage

labels which could provide enough knowledge for learning. However, the other three data sets have fewer labels, so a large α will reduce the impact of the common features, and results in an inappropriate absolute probability for every label. Therefore, we can conclude that label dependency should be given more weight when learning on data sets with a larger number of labels, whereas it should be exploited properly when learning on a data set with a relatively small number of labels.

We then compare our approach with other algorithms. α is set as 0.25, the specific results are shown in Table II-V, and the bold item indicates the best algorithm on the corresponding data set. It indicates that our approach

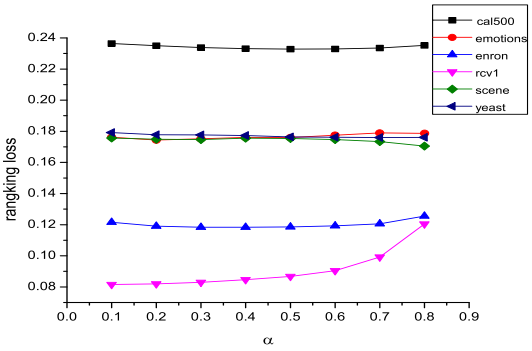

Figure 5. The effect of α in terms of ranking loss

SELD is significantly superior to the other algorithms. For hamming loss, SELD performs best on four data sets, and is only slightly inferior to IBLR-ML+ and SDN-LR on one data set. Compared to the other algorithms, its performance is improved by 7.20% (CC), 10.38% (IBLR-ML+), and 8.47% (CDN-LR) on average for all data sets. For one-error, a similar finding is observed, and SELD outperforms the others by 19.36% (CC), 12.20% (IBLR-ML+), and 13.25% (CDN-LR) averaged on all data sets. For coverage, SELD performs best on five data sets, and is only slightly inferior to IBLR-ML+ on data set *scene*. It outperforms the other algorithms on average by 25.50% (CC), 25.18% (IBLR-ML+), and 18.96% (CDN-LR). For ranking loss, it shows similar performance. SELD performs best on four data sets, and is only slightly inferior to IBLR on two data sets. The prediction performance is increased by 33.70%, 25.65%, and 19.88% in comparison to CC, IBLR-ML+, and CDN-LR, respectively.

We notice that our approach shows much better performance on data sets: *cal500*, *enron*, and *rcv1* in particular. We have known that these three data sets all have a larger number of labels, so they would exhibit complex label dependencies, which is an advantage to our algorithm. In summary, our proposed method has shown the distinguished superiority in predicting labels especially on data sets with a relatively larger number of labels.

VI. Conclusions

In this paper, we proposed a new paradigm of exploiting label dependencies for multi-label learning. It employs the random walk with restart strategy to exploit dependencies between labels in a supervised way. Our method's advantages include: (1) instead of adding dependent labels into the feature set to train prediction models, we separated them into two exclusive components, thus we can give a flexible weight to the set of dependent labels and the impact of label dependencies can be intensified; (2) label dependency is exploited in an iterative way, so a globally optimal result can be achieved since the labels' predictions are influenced mutually and updated simultaneously, and finally every label's

Table II
PERFORMANCE IN TERMS OF HAMMING LOSS

Dataset	SELD	CC	IBLR-ML+	CDN-LR
cal500	**0.1871**	0.2181	0.2318	0.1981
emotions	0.2226	0.2465	0.2271	**0.2218**
enron	**0.1092**	0.1218	0.1096	0.1109
rcv1	**0.0424**	0.0486	0.0482	0.0440
scene	0.2093	0.1503	**0.1326**	0.1489
yeast	**0.1450**	0.2264	0.2000	0.2119

Table III
PERFORMANCE IN TERMS OF ONE-ERROR

Dataset	SELD	CC	IBLR-ML+	CDN-LR
cal500	**0.6412**	0.8725	0.8547	0.9163
emotions	0.3019	0.3575	0.3120	**0.2868**
enron	**0.3795**	0.7103	0.6533	0.6780
rcv1	**0.6262**	0.7100	0.704	0.6832
scene	0.3984	0.3993	**0.3594**	0.3972
yeast	**0.2470**	0.2789	0.2573	0.2524

Table IV
PERFORMANCE IN TERMS OF COVERAGE

Dataset	SELD	CC	IBLR-ML+	CDN-LR
cal500	**0.7527**	0.9515	0.9749	0.8537
emotions	**0.3067**	0.3448	0.3811	0.3148
enron	**0.2741**	0.6093	0.5728	0.6103
rcv1	**0.1730**	0.3555	0.4682	0.2915
scene	0.1607	0.1653	**0.1226**	0.1632
yeast	**0.4635**	0.5264	0.6153	0.4733

Table V
PERFORMANCE IN TERMS OF RANKING LOSS

Dataset	SELD	CC	IBLR-ML+	CDN-LR
cal500	**0.2344**	0.3712	0.4354	0.2576
emotions	**0.1748**	0.2172	0.1876	0.1794
enron	**0.1189**	0.3337	0.3242	0.3296
rcv1	**0.0825**	0.1916	0.1526	0.1385
scene	0.1756	0.1811	**0.1671**	0.1771
yeast	0.1776	0.2179	**0.1720**	0.1822

prediction converges to the globally stable status; and (3) The process of exploiting label dependency is implemented in a supervised way to optimize a specific loss function.

We conducted extensive experiments over a broad range of data sets. Firstly, the results shown that it is more suitable to exploit label fully when dealing with data sets which have a large number of labels. Secondly, The results also show that our approach gains significant improvements against other state-of-the-art approaches which exploit label dependency in the phase of training models. It justifies our claim that our approach can exploit label dependency more effectively.

In this work, only two features are used to depict the dependency between labels. In future, we aim to take other factors into consideration in order to make the obtained label dependencies more accurate and reasonable.

REFERENCES

[1] R. E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2, pp. 135–168, 2000.

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

[3] Y. Song, L. Zhang, and C. L. Giles, "Automatic tag recommendation algorithms for social recommender systems," *ACM Transactions On the Web*, vol. 1, no. 5, p. 4, 2011.

[4] K. Dembczynski, W. Waegeman, and W. Cheng, "On label dependence and loss minimization in multi-label classification," *Machine Learning*, vol. 88, no. 1-2, pp. 5–45, 2012.

[5] W. Cheng and E. Hullermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Machine Learning*, vol. 76, no. 2-3, pp. 211–225, 2009.

[6] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 333–359, 2011.

[7] K. Dembczynski, W. Cheng, and E. Hullermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proceedings of the 27th International Conference on Machine Learning*, Aifa, Israel, 2010, pp. 279–286.

[8] M. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2010, pp. 999–1008.

[9] Y. Guo and S. Gu, "Multi-label classification using conditional dependency networks," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, Barcelona, Catalonia, Spain, 2011, pp. 1300–1305.

[10] C. Bielza, G. Li, and P. Larranage, "Multi-dimensional classification with bayesian networks," *International Journal of Approximate Reasoning*, vol. 52, no. 6, pp. 705–727, 2011.

[11] T. N. Rubin, A. Chambers, and P. Smyth, "Statistical topic models for multi-label document classification," *Machine Learning*, vol. 88, no. 1-2, pp. 157–208, 2012.

[12] S.-H. Park and J. Furnkranz, "Multi-label classification with label constraints," in *Proceedings of the ECML PKDD 2008 Workshop on Preference Learning*, Antwerp, Belgium, 2008, pp. 157–171.

[13] W. Feng and J. Wang, "Incorporating heterogeneous information for personalized tag recommendation in social tagging systems," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, 2012, pp. 1276–1284.

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