LIFT: Multi-Label Learning with Label-Specific Features

Min-Ling Zhang, *Member, IEEE* and Lei Wu

Abstract—Multi-label learning deals with the problem where each example is represented by a single instance (feature vector) while associated with a set of class labels. Existing approaches learn from multi-label data by manipulating with *identical* feature set, i.e. the very instance representation of each example is employed in the discrimination processes of all class labels. However, this popular strategy might be suboptimal as each label is supposed to possess specific characteristics of its own. In this paper, another strategy to learn from multi-label data is studied, where *label-specific* features are exploited to benefit the discrimination of different class labels. Accordingly, an intuitive yet effective algorithm named LIFT, i.e. *multi-label learning with Label specIfic FeaTures*, is proposed. LIFT firstly constructs features specific to each label by conducting clustering analysis on its positive and negative instances, and then performs training and testing by querying the clustering results. Comprehensive experiments on a total of seventeen benchmark data sets clearly validate the superiority of LIFT against other well-established multi-label learning algorithms as well as the effectiveness of label-specific features.

Index Terms—machine learning, multi-label learning, label correlations, label-specific features

1 INTRODUCTION

Multi-label learning aims to build classification models for objects assigned with *multiple* class labels simultaneously [45]. Multi-label objects widely exist in various real-world applications, such as text categorization where a news document could cover several topics including *politics*, *economics*, and *reform* [31], [39], [40], multimedia content annotation where one image could demonstrate several scenes including *beach* and *building* [3], [4], [49], bioinformatics where one gene could have a number of functionalities including *metabolism*, *protein synthesis*, and *transcription* [1], [5], [13].

Formally, let $\mathcal{X} = \mathbb{R}^d$ denote the d-dimensional input space and $\mathcal{Y} = \{l_1, l_2, \cdots, l_q\}$ denote the label space with q class labels. Then, the task of multi-label learning is to derive a multi-label classification function $h : \mathcal{X} \to 2^{\mathcal{Y}}$ which assigns each instance $\boldsymbol{x} \in \mathcal{X}$ with a set of relevant labels $h(\boldsymbol{x}) \subseteq \mathcal{Y}$. In recent years, significant amount of learning approaches have been proposed to dealing with multi-label data [59]. One common strategy adopted by existing approaches is that all the class labels in $\mathcal{Y}$ are discriminated based on *identical* feature representation of the instance, i.e. $\boldsymbol{x}$. In other words, a family of q functions $\{f_1, f_2, \cdots, f_q\}$ are induced from the multi-label training examples, where each function $f_k : \mathcal{X} \to \mathbb{R}$ determines the class membership of l_k to each instance, i.e. $h(\boldsymbol{x}) = \{l_k \mid f_k(\boldsymbol{x}) > 0, 1 \leq k \leq q\}$. Here, each function in the family manipulates with the same feature set $\boldsymbol{x}$.

Although the above strategy has been successful in designing many multi-label learning algorithms [59], it might be suboptimal as each class label is supposed to possess specific characteristics of its own. For example, in automatic image annotation, *color*-based features would be preferred in discriminating sky and non-sky images, *texture*-based features would be preferred in discriminating desert and non-desert images, while both *color*- and *texture*-based features might be useful in discriminating other labels in the label space. For another example, in text categorization, features related to word terms such as *government*, *national security* and *presidential election* would be informative in discriminating political and non-political documents, while features related to word terms such as *GDP*, *tax reduction* and *stock markets* would be informative in discriminating economic and non-economic documents.

Therefore, we hypothesize that if *label-specific* features, i.e. the most pertinent and discriminative features for each class label, could be used in the learning process, a more effective approach to learning from multi-label data could be achieved. In this paper, a new algorithm named LIFT, i.e. multi-label learning with *Label specIfic FeaTures*, is proposed. Briefly, LIFT learns from multi-label data with two intuitive steps. Firstly, for each class label $l_k \in \mathcal{Y}$, clustering analysis is performed on its positive as well as negative training instances,and then features specific to l_k are constructed by querying the clustering results. Secondly, a family of q classifiers are induced with each of them being derived from the generated label-specific features other than the original ones.

To thoroughly evaluate the performance of the proposed approach, comparative studies over seventeen regular-scale and large-scale data sets have been conducted. Experimental results show that: (a) LIFT achieves

- *Min-Ling Zhang and Lei Wu are with the School of Computer Science and Engineering, Nanjing 210096, China, and the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China. Email: {zhangml,wul}@seu.edu.cn*

superior performance against several state-of-the-art multi-label learning algorithms; (b) LIFT's label-specific features have the potential of being a general strategy to improve multi-label learning algorithms comprising a number of binary classifiers; (c) LIFT's label-specific features are highly comparable to other feature manipulation mechanisms including feature selection [17], meta-level features [52] and shared subspace [23], [24].

The rest of this paper is organized as follows. Section 2 briefly reviews existing approaches to multi-label learning. Section 3 presents the proposed LIFT approach. Section 4 reports comparative experimental results over a wide range of multi-label data sets. Finally, Section 5 concludes and discusses several issues for future work.

2 Related Work

Early researches on multi-label learning originate from the investigation of multi-label text categorization techniques [31], [40], [46]. In recent years, multi-label learning has drawn much attentions from machine learning and related communities, which has been successfully applied to diverse applications such as bioinformatics [1], [5], multimedia contents analysis including image [3], [4], [49], audio [27], [34], [42], and video [35], [41], [48], web mining [25], [33], [36], information retrieval [52], [54], etc. Accordingly, significant amount of algorithms have been proposed to learning from multi-label data [45], [59].

Generally speaking, the task of inducing multi-label classification functions is challenging as the classifier's output space is exponential in size to the number of possible class labels (i.e. $|2^{\mathcal{Y}}| = 2^q$). To cope with this issue, one common strategy adopted by existing approaches is to exploit label *correlations* to facilitate the learning process. Based on the order of correlations being considered, existing approaches can be roughly grouped into three major categories [56], namely *first-order* approaches, *second-order* approaches and *high-order* approaches.

First-order approaches tackle multi-label learning problem by decomposing it into a number of *independent* binary classification problems. Here, one function f_k is learned for each possible class label l_k $(1 \leq k \leq q)$ by ignoring the co-existence of other labels l_j $(j \neq k)$. For any multi-label training example $(\boldsymbol{x}, Y)$ $(\boldsymbol{x} \in \mathcal{X}, Y \subseteq \mathcal{Y})$, $\boldsymbol{x}$ will be regarded as one positive example to learn f_k if $l_k \in Y$, and one negative example otherwise. Based on those training examples, f_k can be induced with popular learning techniques such as AdaBoost [40], k-nearest neighbor [58], kernel methods [3], decision trees [8], [9], etc. The major advantage of first-order approaches lies in their conceptual simplicity and high efficiency. While on the other hand, these approaches could be less effective due to their ignorance of label correlations.

Second-order approaches tackle multi-label learning problem by exploiting *pairwise* relationships between the labels. One way to consider pairwise relationship is to impose the ranking criterion that for any multi-label training example $(\boldsymbol{x}, Y)$, given its pair of relevant label $l_k \in Y$ and irrelevant label $l_{k'} \notin Y$, f_k should yield larger output than $f_{k'}$ on $\boldsymbol{x}$. The ranking criterion can be incorporated into the objective function to be optimized by learning models such as support vector machines [13], neural networks [28], [57]. Another way to consider pairwise relationship is to exploit the co-occurrence patterns over each of the $\binom{q}{2}$ label pairs (l_k, l_j) $(k \neq j)$. Label co-occurrence patterns can be adopted as expectation constraints for maximum entropy classifiers [15], [60], or be utilized to decompose multi-label learning problem into $\binom{q}{2}$ pairwise comparison problems [14], [29]. Second-order approaches address label correlations to certain extent and thus are relatively effective. However, in real-world scenarios label correlations could be rather complex and go beyond second-order.

High-order approaches tackle multi-label learning problem by exploring *high-order* relationships among the labels. One straightforward choice is to model interactions among all class labels, i.e. to consider all other labels' influences on each label. This choice can be accomplished by assuming linear combination [7], nonlinear mapping [16], [32], or shared subspace [24], [51] over the whole label space. Another choice is to model interactions among a subset of class labels instead of all of them. This choice can be accomplished by selecting the label subsets randomly [26], [38], [43] or by determining the label subsets specified by graph structure [2], [18], [56]. Apparently, high-order approaches have stronger correlation-modeling capabilities than the first-order and second-order counterparts. Nevertheless, these approaches would be more computationally demanding and less scalable.

As reviewed above, existing approaches have the common property of handling multi-label data by focusing on the output space, while *identical* feature set inherited from the original input space is employed in discriminating all the class labels. In the next section, a new approach named LIFT is proposed which handles multi-label data by focusing on the input space via label-specific features.

3 The LIFT Approach

3.1 Algorithmic Details

Given a set of m multi-label training examples $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq m\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is a d-dimensional feature vector and $Y_i \subseteq \mathcal{Y}$ is the set of relevant labels associated with $\boldsymbol{x}_i$. Then, LIFT learns from $\mathcal{D}$ by taking two elementary steps, i.e. *label-specific features construction* and *classification models induction*.

In the *first* step, LIFT aims to construct features which could effectively capture the specific characteristics of each label, so as to provide appropriate distinguishing information to facilitate its discrimination process. To achieve this, it is necessary to investigate the underlying properties of the training instances with respect to each

class label. Specifically, for one class label $l_k \in \mathcal{Y}$, the set of positive training instances $\mathcal{P}_k$ as well as the set of negative training instances $\mathcal{N}_k$ correspond to:

$$\begin{aligned} \mathcal{P}_k &= \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_k \in Y_i\} \\ \mathcal{N}_k &= \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_k \notin Y_i\} \end{aligned} \tag{1}$$

In other words, $\mathcal{P}_k$ and $\mathcal{N}_k$ consist of the training instances with and without label l_k respectively.

To gain insights on the properties of $\mathcal{P}_k$ and $\mathcal{N}_k$, LIFT chooses to employ *clustering techniques* which have been widely used as stand-alone tools for data analysis. In this paper, the popular k-means algorithm [22] is adopted due to its effectiveness and simplicity. Therefore, suppose $\mathcal{P}_k$ is partitioned into m_k^+ disjoint clusters whose centers are denoted as $\left\{\boldsymbol{p}_1^k, \boldsymbol{p}_2^k, \cdots, \boldsymbol{p}_{m_k^+}^k\right\}$. Similarly, $\mathcal{N}_k$ is also partitioned into m_k^- disjoint clusters whose centers are denoted as $\left\{\boldsymbol{n}_1^k, \boldsymbol{n}_2^k, \cdots, \boldsymbol{n}_{m_k^-}^k\right\}$. Multi-label learning tasks usually encounter the issue of *class-imbalance* [59], where the number of positive instances for each class label is much smaller than the number of negative ones, i.e. $|\mathcal{P}_k| \ll |\mathcal{N}_k|$. To mitigate potential risks brought by the class-imbalance problem, LIFT sets equivalent number of clusters for $\mathcal{P}_k$ and $\mathcal{N}_k$, i.e. $m_k^+ = m_k^- = m_k$. In this way, clustering information gained from positive instances as well as negative instances are treated with equal importance.

Specifically, the number of clusters retained for $\mathcal{P}_k$ and $\mathcal{N}_k$ is set as follows:

$$m_k = \lceil r \cdot \min(|\mathcal{P}_k|, |\mathcal{N}_k|) \rceil \tag{2}$$

Here, $|\cdot|$ returns the set cardinality and $r \in [0, 1]$ is a ratio parameter controlling the number of clusters being retained.

Conceptually, cluster centers identified by the k-means algorithm characterize the underlying structure of the training instances with regard to l_k, which can be served as appropriate building blocks (prototypes) for the construction of label-specific features. Here, a mapping $\phi_k : \mathcal{X} \to \mathcal{Z}_k$ from the original d-dimensional input space $\mathcal{X}$ to the $2m_k$-dimensional label-specific feature space is created as follows:[1]

$$\begin{aligned} &\phi_k(\boldsymbol{x}) = \\ &\left[d(\boldsymbol{x}, \boldsymbol{p}_1^k), \cdots, d(\boldsymbol{x}, \boldsymbol{p}_{m_k}^k), d(\boldsymbol{x}, \boldsymbol{n}_1^k), \cdots, d(\boldsymbol{x}, \boldsymbol{n}_{m_k}^k)\right] \end{aligned} \tag{3}$$

Here, $d(\cdot, \cdot)$ returns the distance between two instances and is set to Euclidean metric in this paper.

In the *second* step, a family of q classification models $\{g_1, g_2, \cdots, g_q\}$ are induced with the generated label-specific features. Here, for each class label $l_k \in \mathcal{Y}$, a new *binary* training set $\mathcal{B}_k$ with m examples is created from the original multi-label training set $\mathcal{D}$ and the mapping ϕ_k as follows:

$$\begin{aligned} &\mathcal{B}_k = \{(\phi_k(\boldsymbol{x}_i), Y_i(k)) \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}\} \text{ where} \\ &Y_i(k) = +1 \text{ if } l_k \in Y_i; \text{ Otherwise, } Y_i(k) = -1 \end{aligned} \tag{4}$$

Based on $\mathcal{B}_k$, any binary learner $\mathfrak{L}$ can be applied to induce a classification model $g_k : \mathcal{Z}_k \to \mathbb{R}$ for l_k.

Given an unseen example $\boldsymbol{u} \in \mathcal{X}$, its associated label set is predicted as:

$$Y = \{l_k \mid g_k(\phi_k(\boldsymbol{u})) > 0, 1 \le k \le q\} \tag{5}$$

In other words, classification model f_k corresponding to each label l_k can be viewed as the composition of g_k and ϕ_k, i.e. $f_k(\boldsymbol{u})=[g_k \circ \phi_k](\boldsymbol{u}) = g_k(\phi_k(\boldsymbol{u}))$.

Table 1 illustrates the complete procedure of LIFT. Given the multi-label training examples, LIFT firstly constructs label-specific features for each possible class label (steps 1 to 5); After that, a family of q binary classification models are induced based on the constructed features successively (steps 6 to 9); Finally, the unseen example is fed to the learned models for prediction (step 10).

TABLE 1
Pseudo-code of LIFT.

Y =LIFT($\mathcal{D}$, r, $\mathfrak{L}$, $\boldsymbol{u}$)
Inputs:
$\mathcal{D}$: multi-label training set $\{(\boldsymbol{x}_i, Y_i) \mid 1 \le i \le m\}$ ($\boldsymbol{x}_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y}, \mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{l_1, l_2, \cdots, l_q\}$)
r : ratio parameter as used in Eq.(2)
$\mathfrak{L}$: binary learner for classifier induction
$\boldsymbol{u}$: unseen example ($\boldsymbol{u} \in \mathcal{X}$)
Outputs:
Y : predicted label set for $\boldsymbol{u}$ ($Y \subseteq \mathcal{Y}$)
Process:
1. **for** $k = 1$ to q **do**
2. Form $\mathcal{P}_k$ and $\mathcal{N}_k$ based on $\mathcal{D}$ according to Eq.(1);
3. Perform k-means clustering on $\mathcal{P}_k$ and $\mathcal{N}_k$, each with m_k clusters as defined in Eq.(2);
4. Create the mapping ϕ_k for l_k according to Eq.(3);
5. **endfor**
6. **for** $k = 1$ to q **do**
7. Form $\mathcal{B}_k$ according to Eq.(4);
8. Induce g_k by invoking $\mathfrak{L}$ on $\mathcal{B}_k$, i.e. $g_k \leftarrow \mathfrak{L}(\mathcal{B}_k)$;
9. **endfor**
10. Return Y according to Eq.(5)

3.2 Remarks

In terms of label-specific features construction, the process shown in Table 1 (steps 1 to 5) only represents an *intuitive* implementation and is not meant to be the best possible practice for constructing label-specific

1. As discussed above, the distribution of positive instances and negative instances for each label is usually *imbalanced* with $|\mathcal{P}_k| \ll |\mathcal{N}_k|$. Thus, the dimensionality of $\mathcal{Z}_k$, i.e. $2 \cdot \lceil r \cdot \min(|\mathcal{P}_k|, |\mathcal{N}_k|) \rceil$, would be of reasonable size in most cases. As an instance, for the *bibtex* data set with 7395 examples and 1836 features (Table 2), the dimensionality of the label-specific feature space is only around 23 ± 20 across all labels with $r = 0.1$.

TABLE 2
Characteristics of the experimental data sets.

Data set	$\|\mathcal{S}\|$	$dim(\mathcal{S})$	$L(\mathcal{S})$	$F(\mathcal{S})$	$LCard(\mathcal{S})$	$LDen(\mathcal{S})$	$DL(\mathcal{S})$	$PDL(\mathcal{S})$	Domain	URL*
CAL500	502	68	174	numeric	26.044	0.150	502	1.000	music	URL 1
language log	1460	1004	75	nominal	1.180	0.016	286	0.196	text	URL 2
enron	1702	1001	53	nominal	3.378	0.064	753	0.442	text	URL 2
image	2000	294	5	numeric	1.236	0.247	20	0.010	images	URL 3
scene	2407	294	6	numeric	1.074	0.179	15	0.006	images	URL 1
yeast	2417	103	14	numeric	4.237	0.303	198	0.082	biology	URL 3
slashdot	3782	1079	22	nominal	1.181	0.054	156	0.041	text	URL 2
corel5k	5000	499	374	nominal	3.522	0.009	3175	0.635	images	URL 1
rcv1 (subset 1)	6000	944	101	numeric	2.880	0.029	1028	0.171	text	URL 1
rcv1 (subset 2)	6000	944	101	numeric	2.634	0.026	954	0.159	text	URL 1
bibtex	7395	1836	159	nominal	2.402	0.015	2856	0.386	text	URL 1
corel16k (sample 1)	13766	500	153	nominal	2.859	0.019	4803	0.349	images	URL 1
corel16k (sample 2)	13761	500	164	nominal	2.882	0.018	4868	0.354	images	URL 1
eurlex(subject matter)	19348	5000	201	numeric	2.213	0.011	2504	0.129	text	URL 1
eurlex(directory code)	19348	5000	412	numeric	1.292	0.003	1615	0.084	text	URL 1
tmc2007	28596	981	22	nominal	2.158	0.098	1341	0.047	text	URL 1
mediamill	43907	120	101	numeric	4.376	0.043	6555	0.149	video	URL 1

* URL 1: http://mulan.sourceforge.net/datasets.html
URL 2: http://meka.sourceforge.net/#datasets
URL 3: http://cse.seu.edu.cn/people/zhangml/Resources.htm#data

features. Actually, the mapping ϕ_k can be implemented in numerous alternative ways, such as setting different number of clusters for positive and negative instances (i.e. $m_k^+ \neq m_k^-$), utilizing more sophisticated distance for $d(\cdot,\cdot)$ other than the Euclidean metric, or even employing kNN rule [47] to identify the prototypes for feature mapping other than invoking the k-means procedure, etc.[2]; In terms of classification models induction, the process shown in Table 1 (steps 6 to 9) is similar to those of the *first-order* approaches as discussed in Section 2. The major difference lies that LIFT induces the classifier on l_k with label-specific feature set $\phi_k(\boldsymbol{x})$ instead of the original feature set $\boldsymbol{x}$.

In general sense, LIFT embodies three major merits that any practically useful algorithm is desirable to have: 1) *Flexibility*: Besides the label-specific features which could be generated in various ways as discussed above, the classification models could also be induced with various binary learners $\mathfrak{L}$ to meet different requirements (e.g. $\mathfrak{L}$="decision tree" for low training cost; $\mathfrak{L}$="rule learner" for good comprehensibility, etc.); 2) *Simplicity*: As shown in Table 1, the LIFT algorithm is succinct and easy to implement. Specifically, LIFT is affiliated with only *one single* parameter (i.e. r) which keeps the algorithm away from the sophisticated (and often tricky) issue of tuning a number of parameters simultaneously; 3) *Effectiveness*: As to be reported in Section 4, LIFT shows highly competitive performance against the state-of-the-art multi-label learning methods. To convincingly validate the effectiveness of LIFT, a total of seventeen benchmark multi-label data sets have been employed for experimental studies.

2. Several variants of LIFT have been studied in Subsection 4.4.2.

4 EXPERIMENTS

4.1 Experimental Configuration

4.1.1 Data Sets

For each data set $\mathcal{S} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq p\}$, we use $|\mathcal{S}|$, $dim(\mathcal{S})$, $L(\mathcal{S})$ and $F(\mathcal{S})$ to denote the *number of examples*, *number of features*, *number of possible class labels*, and *feature type* for $\mathcal{S}$ respectively. In addition, several other multi-label properties [38], [45] are denoted as:

- $LCard(\mathcal{S}) = \frac{1}{p}\sum_{i=1}^{p}|Y_i|$: *label cardinality* which measures the average number of labels per example;
- $LDen(\mathcal{S}) = \frac{LCard(\mathcal{S})}{L(\mathcal{S})}$: *label density* which normalizes $LCard(\mathcal{S})$ by the number of possible labels;
- $DL(\mathcal{S}) = |\{Y|(\boldsymbol{x}, Y) \in \mathcal{S}\}|$: *distinct label sets* which counts the number of distinct label combinations in $\mathcal{S}$;
- $PDL(\mathcal{S}) = \frac{DL(\mathcal{S})}{|\mathcal{S}|}$: *proportion of distinct label sets* which normalizes $DL(\mathcal{S})$ by the number of examples.

Table 2 summarizes detailed characteristics of all multi-label data sets used in the experiments. Roughly ordered by $|\mathcal{S}|$, eight regular-scale data sets (first part, $|\mathcal{S}| \leq 5000$) as well as nine large-scale data sets (second part, $|\mathcal{S}| > 5000$) are included. Furthermore, dimensionality reduction is performed on three text data sets with huge number of features ($dim(\mathcal{S}) > 47000$), including *rcv1 (subset 1)*, *rcv1 (subset 2)*, and *tmc2007*. Specifically, the top 2% features with highest *document frequency* [53] are retained.[3]

3. As a routine process, stop words frequently appearing in the document (e.g. the function words) are not included in the vocabulary. Furthermore, existing studies show that based on document frequency, no loss will be incurred by retaining 10% features and just a small loss will be incurred by retaining 1% features [53], [57].

As shown in Table 2, the seventeen data sets cover a broad range of cases with diversified multi-label properties. To the best of our knowledge, few works on multi-label learning has conducted experimental evaluation across such broad range of data sets. Few notable exceptions are [30] and [38] where a total of 11 and 15 data sets have been considered respectively. Therefore, experimental studies reported in this paper are quite comprehensive which aim to provide a solid basis for thorough evaluation of LIFT's effectiveness.[4]

4.1.2 Evaluation Metrics

Performance evaluation on multi-label learning algorithms is somewhat complicated as each object is associated with multiple class labels simultaneously, where traditional single-label criteria such as *accuracy*, *precision*, *recall*, etc. can not be directly applied [59]. Given the test data set $\mathcal{T} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq t\}$ and the family of q learned functions $\{f_1, f_2, \cdots, f_q\}$, five evaluation metrics widely-used in multi-label learning [45], [59] are employed in this paper:

• *Hamming loss*:

$$\text{hloss} = \frac{1}{t}\sum_{i=1}^{t}|h(\boldsymbol{x}_i)\Delta Y_i|$$

Here, $h(\boldsymbol{x}_i) = \{l_k \mid f_k(\boldsymbol{x}_i) > 0, 1 \leq k \leq q\}$ corresponds to the predicted set of relevant labels for $\boldsymbol{x}_i$, and Δ stands for the symmetric difference between two sets. Hamming loss evaluates the fraction of instance-label pairs which have been misclassified, i.e. a relevant label is missed or an irrelevant label is predicted.

• *One-error*:

$$\text{one-error} = \frac{1}{t}\sum_{i=1}^{t}[\![[\arg\max_{l_k \in \mathcal{Y}} f_k(\boldsymbol{x}_i)] \notin Y_i]\!]$$

Here, for any predicate π, $[\![\pi]\!]$ returns 1 if π holds and 0 otherwise. One-error evaluates the fraction of examples whose top-ranked predicted label is not in the ground-truth relevant label set.

• *Coverage*:

$$\text{coverage} = \frac{1}{q}\left(\frac{1}{t}\sum_{i=1}^{t}\max_{l_k \in Y_i} rank(\boldsymbol{x}_i, l_k) - 1\right)$$

Here, $rank(\boldsymbol{x}_i, l_k) = \sum_{j=1}^{q}[\![f_j(\boldsymbol{x}_i) \geq f_k(\boldsymbol{x}_i)]\!]$ returns the rank of l_k when all class labels in $\mathcal{Y}$ are sorted in descending order according to $\{f_1(\boldsymbol{x}_i), f_2(\boldsymbol{x}_i), \cdots, f_q(\boldsymbol{x}_i)\}$. Coverage evaluates how many steps are needed, on average, to move down the ranked label list of an example so as to cover all its relevant labels. Furthermore, the coverage metric is normalized by the number of possible class labels (i.e. q) in this paper.

• *Ranking loss*:

$$\text{rloss} = \frac{1}{t}\sum_{i=1}^{t}\frac{|\{(l_k, l_j) \mid f_k(\boldsymbol{x}_i) \leq f_j(\boldsymbol{x}_i), (l_k, l_j) \in Y_i \times \bar{Y}_i\}|}{|Y_i||\bar{Y}_i|}$$

Here, $\bar{Y}_i$ is the complementary set of Y_i in $\mathcal{Y}$. Ranking loss evaluates the average fraction of misordered label pairs, i.e. an irrelevant label of an example is ranked higher than its relevant one.

• *Average precision*:

$$\text{avgprec} = \frac{1}{t}\sum_{i=1}^{t}\frac{1}{|Y_i|}\sum_{l_k \in Y_i}\frac{|\mathcal{R}(\boldsymbol{x}_i, l_k)|}{rank(\boldsymbol{x}_i, l_k)}, \quad \text{where}$$

$$\mathcal{R}(\boldsymbol{x}_i, l_k) = \{l_j \mid rank(\boldsymbol{x}_i, l_j) \leq rank(\boldsymbol{x}_i, l_k), l_j \in Y_i\}$$

Average precision evaluates the average fraction of relevant labels ranked higher than a particular label $l_k \in Y_i$.

The above metrics are *example-based* ones which work by evaluating the classification models' performance on each test example separately, and then returning the averaged value across the test set. Specifically, *hamming loss* considers how $\{f_1, f_2, \cdots, f_q\}$ perform in terms of *classification* quality, while the other four metrics considers how they perform in terms of *ranking* quality.

In addition to example-based metrics, *label-based* metrics work by evaluating the binary classification performance on each class label separately, and then returning the averaged value across all class labels. In this paper, we choose to employ the AUC criterion (area under the ROC curve) for performance evaluation on each class label:[5]

• *Macro-averaging AUC*:

$$\text{AUC}_{macro} = \frac{1}{q}\sum_{k=1}^{q}\text{AUC}_k$$

$$= \frac{1}{q}\sum_{k=1}^{q}\frac{|(\boldsymbol{x}', \boldsymbol{x}'') \mid f_k(\boldsymbol{x}') \geq f_k(\boldsymbol{x}''), (\boldsymbol{x}', \boldsymbol{x}'') \in \mathcal{P}_k \times \mathcal{N}_k|}{|\mathcal{P}_k||\mathcal{N}_k|}$$

Here, the AUC value on each class label (i.e. AUC_k) is calculated based on the relationship between AUC and the Wilcoxon-Mann-Whitney statistic [20].

Note that for all the six multi-label metrics, their values vary between [0,1]. Furthermore, for average precision and macro-averaging AUC, the *larger* the values the better the performance; While for the other four metrics, the *smaller* the values the better the performance. These metrics serve as good indicators for comprehensive comparative studies as they evaluate the performance of the learned models from various aspects.

4. For brevity, in the rest of this paper we use *rcv1-s1*, *rcv1-s2*, *corel16k-s1*, *corel16k-s2*, *eurlex-sm* and *eurlex-dc* to rename data sets rcv1 (subset 1), rcv1 (subset 2), corel16k (sample 1), corel16k (sample 2), eurlex (subject matter), and eurlex (directory code) respectively.

5. Here, it is possible to employ other single-label criteria such as *accuracy*, *precision*, *recall*, etc. to yield the label-based metric. In this paper, AUC criterion is used due to its capability of evaluating binary classification performance in a more comprehensive way.

TABLE 3
Predictive performance of each comparing algorithm (mean±std. deviation) on the eight regular-scale data sets.

Comparing algorithm	*Hamming loss* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	**0.137±0.002**	**0.015±0.001**	**0.048±0.001**	**0.163±0.003**	**0.084±0.002**	**0.197±0.002**	**0.040±0.001**	**0.010±0.001**
BR	**0.137±0.002**	0.017±0.001	0.060±0.001	0.185±0.004	0.111±0.003	0.201±0.003	0.049±0.001	0.012±0.001
CLR	**0.137±0.002**	0.018±0.001	0.055±0.001	0.186±0.005	0.112±0.003	0.201±0.003	0.050±0.001	0.011±0.001
ECC	0.182±0.005	0.025±0.001	0.056±0.001	0.218±0.027	0.096±0.003	0.207±0.003	0.056±0.001	0.015±0.001
RAKEL	0.138±0.002	0.017±0.001	0.058±0.001	0.173±0.004	0.096±0.004	0.202±0.003	0.048±0.001	0.012±0.001
Comparing algorithm	*One-error* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	0.126±0.016	**0.694±0.010**	**0.254±0.005**	**0.289±0.012**	**0.212±0.011**	0.229±0.011	0.430±0.013	0.706±0.012
BR	0.362±0.039	0.858±0.009	0.498±0.012	0.406±0.012	0.348±0.007	0.256±0.008	0.501±0.007	0.849±0.008
CLR	**0.121±0.016**	0.756±0.008	0.279±0.010	0.328±0.017	0.255±0.009	**0.228±0.007**	0.436±0.005	0.721±0.007
ECC	0.137±0.021	0.720±0.012	0.293±0.008	0.408±0.069	0.247±0.010	0.244±0.009	**0.418±0.009**	**0.699±0.006**
RAKEL	0.286±0.039	0.838±0.014	0.412±0.016	0.312±0.010	0.247±0.009	0.251±0.008	0.453±0.005	0.819±0.010
Comparing algorithm	*Coverage* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	0.757±0.011	0.183±0.007	0.245±0.007	**0.178±0.007**	**0.071±0.002**	**0.458±0.007**	0.115±0.003	0.313±0.008
BR	0.972±0.001	0.468±0.010	0.595±0.010	0.280±0.008	0.158±0.004	0.641±0.005	0.238±0.005	0.898±0.003
CLR	**0.751±0.008**	**0.155±0.010**	**0.229±0.006**	0.190±0.007	0.083±0.003	0.462±0.005	**0.109±0.003**	**0.267±0.004**
ECC	0.806±0.016	0.309±0.014	0.349±0.014	0.229±0.034	0.084±0.002	0.464±0.005	0.130±0.004	0.562±0.007
RAKEL	0.971±0.001	0.459±0.011	0.523±0.008	0.209±0.009	0.104±0.003	0.558±0.006	0.212±0.005	0.886±0.004
Comparing algorithm	*Ranking loss* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	0.185±0.003	0.181±0.008	0.084±0.003	**0.155±0.007**	**0.069±0.002**	**0.169±0.004**	0.100±0.003	0.131±0.003
BR	0.518±0.008	0.421±0.008	0.308±0.007	0.285±0.009	0.171±0.005	0.315±0.005	0.216±0.005	0.655±0.004
CLR	**0.181±0.002**	**0.121±0.007**	**0.079±0.002**	0.171±0.008	0.083±0.004	0.172±0.004	**0.094±0.003**	**0.114±0.002**
ECC	0.204±0.008	0.367±0.011	0.133±0.004	0.224±0.043	0.085±0.003	0.186±0.003	0.131±0.005	0.292±0.003
RAKEL	0.444±0.005	0.412±0.010	0.241±0.005	0.196±0.008	0.107±0.003	0.245±0.004	0.190±0.005	0.627±0.004
Comparing algorithm	*Average precision* ↑							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	0.492±0.005	**0.388±0.007**	**0.685±0.005**	**0.811±0.007**	**0.875±0.006**	**0.763±0.006**	0.671±0.008	**0.280±0.004**
BR	0.275±0.006	0.178±0.009	0.449±0.011	0.709±0.008	0.771±0.005	0.672±0.005	0.572±0.005	0.101±0.003
CLR	**0.499±0.005**	0.377±0.008	0.675±0.005	0.789±0.009	0.850±0.006	0.758±0.005	0.674±0.003	0.274±0.002
ECC	0.482±0.008	0.316±0.009	0.651±0.006	0.739±0.043	0.853±0.005	0.752±0.006	**0.680±0.006**	0.264±0.003
RAKEL	0.353±0.006	0.197±0.013	0.539±0.006	0.788±0.006	0.843±0.005	0.720±0.005	0.617±0.004	0.122±0.004
Comparing algorithm	*Macro-averaging AUC* ↑							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFT	0.502±0.008	**0.714±0.010**	0.688±0.018	**0.844±0.006**	**0.943±0.002**	**0.673±0.007**	**0.847±0.009**	**0.679±0.006**
BR	0.500±0.001	0.517±0.002	0.579±0.007	0.705±0.007	0.801±0.003	0.565±0.003	0.656±0.009	0.518±0.001
CLR	0.533±0.007	0.676±0.014	**0.698±0.013**	0.816±0.007	0.917±0.004	0.645±0.007	0.833±0.016	0.678±0.005
ECC	0.507±0.005	0.544±0.004	0.646±0.008	0.807±0.030	0.931±0.004	0.646±0.003	0.767±0.010	0.568±0.003
RAKEL	**0.547±0.007**	0.520±0.002	0.596±0.007	0.803±0.005	0.884±0.004	0.614±0.003	0.687±0.011	0.521±0.001

4.2 Comparative Studies

In this subsection, we compare LIFT against several state-of-the-art multi-label learning approaches to validate its predictive performance:

• *Binary relevance* (BR) [3]: This is a *first-order* approach which decomposes the multi-label learning problem into q independent binary classification problems. Binary relevance could be viewed as a plain version of LIFT where the label-specific features $\phi_k(\boldsymbol{x})$ is kept to the original features $\boldsymbol{x}$.

• *Calibrated label ranking* (CLR) [14]: This is a *second-order* approach which transforms the multi-label learning problem into a label ranking problem, where pairwise comparison [21] is employed to train $\binom{q}{2}$ binary classifiers to yield the ranking among labels which are further bi-partitioned via an embedded calibration mechanism.

• *Ensembles of classifier chains* (ECC) [38]: This is a *high-order* approach which transforms the multi-label learning problem into a chain of binary classification problems, where binary classifiers in the chain are successively built upon the predictions of preceding ones. In addition, ensemble learning is employed to address chain order randomness. Here, the ensemble size is set to be 100 to accommodate sufficient number of classifier chains.

• *Random k-labelsets* (RAKEL) [43]: This is a *high-order* approach which transforms the multi-label learning problem into an ensemble of multi-class classification problems, where each multi-class classification problem is generated by applying the label powerset techniques [37], [59] on a randomly chosen k-labelset in $\mathcal{Y}$. Here, the ensemble size is set to be $2q$ with $k = 3$ as suggested in the literature.

The only parameter of LIFT, i.e. ratio r as used in Eq.(2), is set to be 0.1 in this paper.[6] Furthermore, for fair comparison, LIBSVM (with linear kernel) [6] is employed

6. As discussed in Subsection 4.4.1, the performance of LIFT becomes stable as r approaches 0.1.

TABLE 4
Predictive performance of each comparing algorithm (mean±std. deviation) on the nine large-scale data sets.

Comparing algorithm	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
	Hamming loss ↓								
LIFT	**0.026±0.001**	**0.023±0.001**	**0.013±0.001**	**0.019±0.001**	**0.018±0.001**	0.008±0.001	0.003±0.001	**0.061±0.001**	**0.030±0.001**
BR	0.031±0.001	0.028±0.001	0.015±0.001	0.020±0.001	0.019±0.001	**0.007±0.001**	**0.002±0.001**	0.071±0.001	0.031±0.001
CLR	0.029±0.001	0.025±0.001	0.014±0.001	**0.019±0.001**	**0.018±0.001**	0.008±0.001	**0.002±0.001**	0.068±0.001	0.031±0.001
ECC	0.030±0.001	0.024±0.001	0.017±0.001	0.030±0.001	**0.018±0.001**	0.010±0.001	0.003±0.001	0.066±0.001	0.035±0.001
RAKEL	0.031±0.001	0.027±0.001	0.015±0.001	0.020±0.001	0.019±0.001	**0.007±0.001**	**0.002±0.001**	0.068±0.001	0.031±0.001
	One-error ↓								
LIFT	**0.412±0.008**	0.421±0.009	0.407±0.008	**0.691±0.008**	**0.686±0.006**	**0.298±0.007**	**0.459±0.017**	**0.213±0.004**	**0.133±0.003**
BR	0.602±0.011	0.522±0.009	0.559±0.004	0.920±0.006	0.920±0.005	0.460±0.015	0.601±0.008	0.339±0.003	0.200±0.003
CLR	0.421±0.005	**0.418±0.004**	**0.401±0.004**	0.702±0.005	0.697±0.005	0.345±0.010	0.495±0.008	0.242±0.003	0.157±0.002
ECC	0.427±0.008	0.427±0.008	0.404±0.003	0.706±0.006	0.712±0.005	0.346±0.007	0.537±0.013	0.232±0.003	0.150±0.005
RAKEL	0.548±0.014	0.472±0.007	0.506±0.005	0.886±0.007	0.897±0.006	0.447±0.016	0.600±0.009	0.253±0.003	0.181±0.002
	Coverage ↓								
LIFT	0.120±0.002	0.123±0.005	0.149±0.006	0.326±0.003	0.315±0.004	**0.096±0.003**	**0.093±0.003**	0.129±0.001	0.184±0.001
BR	0.448±0.005	0.383±0.006	0.461±0.006	0.673±0.002	0.671±0.001	0.552±0.011	0.428±0.005	0.380±0.003	0.575±0.003
CLR	**0.102±0.002**	**0.106±0.003**	**0.118±0.003**	**0.281±0.002**	**0.267±0.002**	0.099±0.002	0.095±0.002	**0.126±0.001**	**0.142±0.001**
ECC	0.187±0.003	0.206±0.007	0.327±0.008	0.446±0.003	0.436±0.002	0.386±0.010	0.347±0.008	0.173±0.002	0.467±0.009
RAKEL	0.414±0.004	0.353±0.006	0.443±0.006	0.667±0.002	0.666±0.001	0.543±0.012	0.428±0.005	0.279±0.003	0.560±0.002
	Ranking loss ↓								
LIFT	0.048±0.001	0.050±0.002	0.082±0.003	0.165±0.002	0.158±0.002	**0.052±0.002**	**0.078±0.003**	0.051±0.001	0.053±0.001
BR	0.279±0.004	0.251±0.004	0.303±0.004	0.422±0.001	0.424±0.001	0.396±0.011	0.395±0.005	0.216±0.003	0.230±0.001
CLR	**0.040±0.001**	**0.042±0.001**	**0.065±0.002**	**0.146±0.001**	**0.139±0.001**	0.057±0.002	0.082±0.002	**0.050±0.001**	**0.038±0.001**
ECC	0.079±0.002	0.096±0.004	0.192±0.003	0.233±0.002	0.229±0.001	0.263±0.007	0.333±0.008	0.074±0.001	0.179±0.008
RAKEL	0.243±0.004	0.216±0.004	0.286±0.003	0.414±0.002	0.418±0.001	0.388±0.011	0.395±0.005	0.139±0.002	0.222±0.001
	Average precision ↑								
LIFT	0.605±0.003	0.617±0.006	0.542±0.008	**0.311±0.003**	**0.306±0.003**	**0.661±0.006**	**0.579±0.013**	**0.814±0.002**	0.714±0.002
BR	0.383±0.007	0.434±0.005	0.363±0.004	0.085±0.002	0.078±0.002	0.427±0.013	0.388±0.008	0.643±0.002	0.502±0.002
CLR	**0.628±0.003**	**0.641±0.003**	**0.564±0.004**	0.306±0.003	0.303±0.002	0.625±0.007	0.546±0.007	0.798±0.002	**0.731±0.001**
ECC	0.606±0.004	0.616±0.005	0.515±0.004	0.282±0.003	0.276±0.002	0.572±0.007	0.458±0.012	0.787±0.002	0.597±0.014
RAKEL	0.436±0.006	0.487±0.005	0.399±0.004	0.103±0.003	0.092±0.003	0.440±0.013	0.389±0.008	0.735±0.002	0.521±0.001
	Macro-averaging AUC ↑								
LIFT	**0.917±0.004**	**0.901±0.007**	0.902±0.004	0.695±0.002	0.709±0.004	0.812±0.008	0.765±0.007	**0.908±0.001**	0.692±0.004
BR	0.609±0.003	0.599±0.004	0.624±0.002	0.516±0.001	0.519±0.001	0.589±0.005	0.562±0.003	0.724±0.002	0.510±0.001
CLR	0.898±0.005	0.884±0.003	**0.908±0.002**	**0.723±0.003**	**0.739±0.003**	**0.888±0.003**	**0.895±0.005**	0.902±0.001	**0.831±0.002**
ECC	0.777±0.005	0.763±0.005	0.763±0.003	0.627±0.004	0.633±0.002	0.624±0.004	0.582±0.004	0.880±0.002	0.524±0.001
RAKEL	0.637±0.004	0.627±0.004	0.641±0.002	0.523±0.001	0.525±0.001	0.591±0.006	0.562±0.003	0.796±0.002	0.513±0.001

as the binary learner for binary classifier induction for LIFT, BR, CLR and ECC.[7]

Tables 3 and 4 report the detailed experimental results of each comparing algorithm on the regular-scale and large-scale data sets respectively. On each data set, 50% examples are randomly sampled without replacement to form the training set, and the rest 50% examples are used to form the test set. The sampling process is repeated for ten times and the average predictive performance across ten training/testing trials are recorded. For each evaluation metric, "↓" indicates "the smaller the better" while "↑" indicates "the larger the better". Furthermore, the best performance among the four comparing algorithms is shown in boldface.

7. As shown in Table 1, the clustering procedure (step 3) for LIFT is accomplished by invoking the k-means algorithm. Note that better choice other than k-means can be adopted by taking into account peculiarities of the data set, e.g. clustering algorithm based on LSA or PLSA [11] is a popular choice for text data sets with sparse feature representation.

TABLE 5
Summary of the Friedman statistics F_F ($k = 5$, $N = 17$) and the critical value in terms of each evaluation metric (k: # comparing algorithms; N: # data sets).

Evaluation metric	F_F	critical value ($\alpha = 0.05$)
Hamming loss	9.0555	2.5153
One-error	52.1503	
Coverage	256.8024	
Ranking loss	256.8024	
Average precision	115.3636	
Macro-averaging AUC	77.6981	

To perform performance analysis among the comparing algorithms systematically, *Friedman test* [10] is employed here which is widely-accepted as the favorable statistical test for comparisons of multiple algorithms over a number of data sets. Given k comparing algorithms and N data sets, let r_i^j denote the rank of the j-th algorithm on the i-th data set (mean ranks are shared in case of ties). Let $R_j = \frac{1}{N}\sum_{i=1}^{N} r_i^j$ denote the average rank

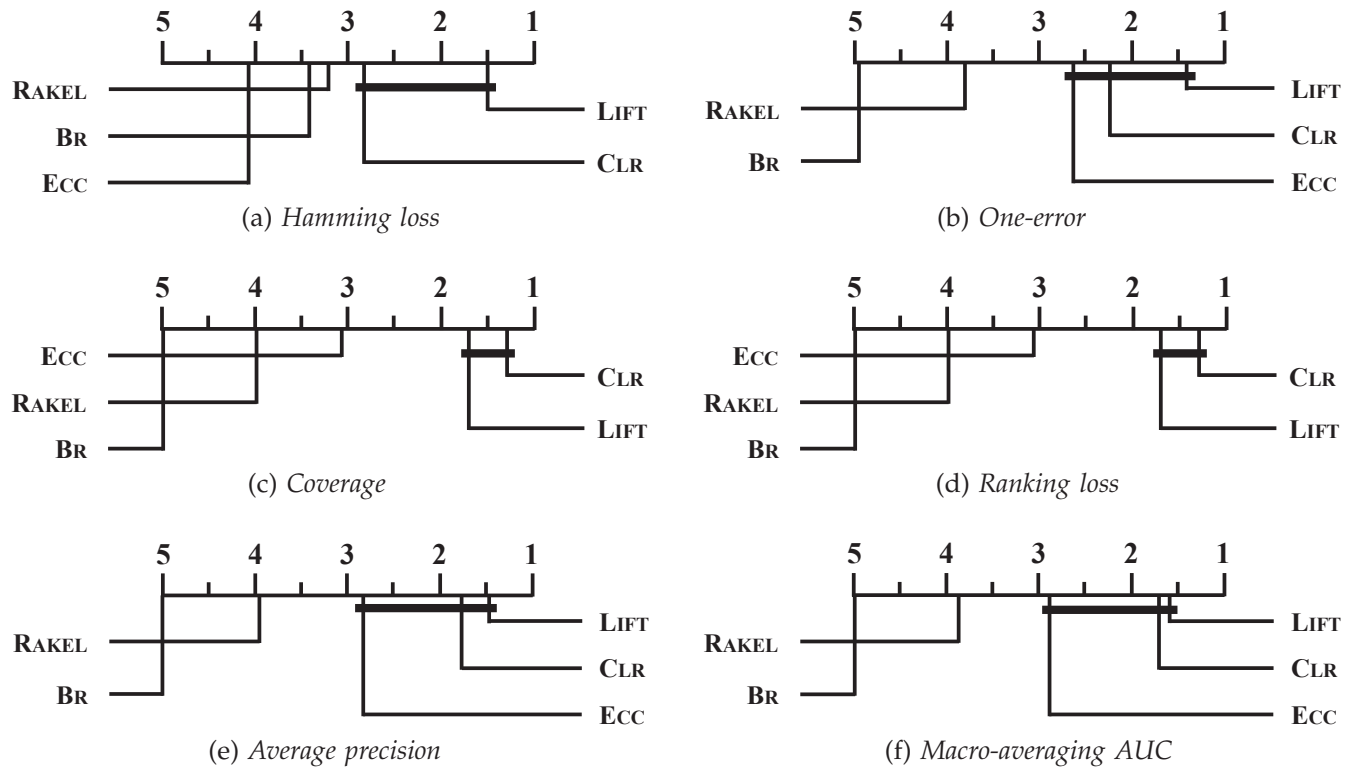


Fig. 1. Comparison of LIFT (control algorithm) against other comparing algorithms with the *Bonferroni-Dunn test*. Algorithms not connected with LIFT in the CD diagram are considered to have significantly different performance from the control algorithm (significance level $\alpha = 0.05$).

for the j-th algorithm, under the null hypothesis (i.e. all algorithms have "equal" performance), the following Friedman statistic F_F will be distributed according to the F-distribution with $k-1$ numerator degrees of freedom and $(k-1)(N-1)$ denominator degrees of freedom:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \quad \text{where}$$
$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

Table 5 summarizes the Friedman statistics F_F and the corresponding critical values on each evaluation metric. As shown in Table 5, at significance level $\alpha = 0.05$, the null hypothesis of "equal" performance among the comparing algorithms is clearly rejected in terms of each evaluation metric. Consequently, we need to proceed with certain *post-hoc test* [10] to further analyze the relative performance among the comparing algorithms. As we are interested in whether the proposed LIFT approach achieves competitive performance against other state-of-the-art approaches, the *Bonferroni-Dunn test* [12] is employed to serve the above purpose by treating LIFT as the control algorithm. Here, the difference between the average ranks of LIFT and one comparing algorithm is compared with the following *critical difference* (CD):

$$\text{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{6}$$

For Bonferroni-Dunn test, we have $q_\alpha = 2.498$ at significance level $\alpha = 0.05$ and thus CD $= 1.3547$ ($k = 5$, $N = 17$). Accordingly, the performance between LIFT and one comparing algorithm is deemed to be significantly different if their average ranks over all data sets differ by at least one CD.

To visually present the relative performance of LIFT and other comparing algorithms, Fig. 1 illustrates the CD diagrams [10] on each evaluation metric, where the average rank of each comparing algorithm is marked along the axis (lower ranks to the right). In each subfigure, any comparing algorithm whose average rank is within one CD to that of LIFT is interconnected with a thick line. Otherwise, any algorithm not connected with LIFT is considered to have significantly different performance between each other.

Based on the above results, the following observations can be made:

1) As shown in Fig. 1, LIFT significantly outperforms BR in terms of each evaluation metric. As BR can be regarded a plain version of LIFT by keeping the original feature vector untouched, the superior performance of LIFT against BR clearly verifies the effectiveness of employing label-specific features.
2) As shown in Fig. 1, LIFT achieves statistically superior or at least comparable performance against ECC and RAKEL in terms of each evaluation metric. Note that ensemble learning strategy has been

TABLE 6
Predictive performance of LIFTed versions of CLR and ECC (mean±std. deviation) on regular-scale (upper part) and large-scale (lower part) data sets.

LIFTed version	*Hamming loss* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.138±0.002	0.016±0.001	0.047±0.001	0.182±0.020	0.085±0.002	0.198±0.002	0.049±0.001	0.010±0.001
LIFTed ECC	0.174±0.006	0.017±0.001	0.053±0.001	0.188±0.020	0.083±0.003	0.208±0.003	0.067±0.003	0.015±0.001
LIFTed version	*One-error* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.115±0.015	0.716±0.011	0.218±0.013	0.315±0.057	0.214±0.012	0.241±0.010	0.442±0.013	0.648±0.012
LIFTed ECC	0.119±0.013	0.820±0.012	0.267±0.019	0.330±0.050	0.213±0.011	0.255±0.008	0.510±0.023	0.688±0.012
LIFTed version	*Coverage* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.744±0.006	0.140±0.008	0.227±0.005	0.195±0.024	0.071±0.002	0.451±0.004	0.125±0.005	0.262±0.004
LIFTed ECC	0.826±0.011	0.380±0.015	0.399±0.015	0.199±0.024	0.077±0.004	0.474±0.007	0.167±0.007	0.707±0.006
LIFTed version	*Ranking loss* ↓							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.179±0.003	0.154±0.008	0.078±0.002	0.188±0.030	0.070±0.003	0.169±0.003	0.137±0.005	0.117±0.002
LIFTed ECC	0.208±0.009	0.468±0.010	0.153±0.004	0.182±0.030	0.076±0.004	0.198±0.005	0.174±0.008	0.431±0.006
LIFTed version	*Average precision* ↑							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.505±0.005	0.377±0.008	0.698±0.008	0.785±0.035	0.874±0.006	0.755±0.006	0.641±0.009	0.313±0.004
LIFTed ECC	0.483±0.010	0.194±0.011	0.646±0.008	0.786±0.031	0.871±0.006	0.741±0.006	0.592±0.016	0.231±0.003
LIFTed version	*Macro-averaging AUC* ↑							
	CAL500	language log	enron	image	scene	yeast	slashdot	corel5k
LIFTed CLR	0.555±0.008	0.685±0.018	0.706±0.014	0.824±0.019	0.933±0.002	0.635±0.007	0.791±0.016	0.697±0.005
LIFTed ECC	0.538±0.007	0.518±0.001	0.604±0.006	0.829±0.023	0.932±0.003	0.615±0.006	0.726±0.010	0.522±0.001

LIFTed version	*Hamming loss* ↓								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.026±0.001	0.023±0.001	0.013±0.001	0.019±0.001	0.017±0.001	0.008±0.001	0.003±0.001	0.060±0.001	0.031±0.001
LIFTed ECC	0.030±0.001	0.027±0.001	0.014±0.001	0.028±0.001	0.026±0.001	0.009±0.001	0.003±0.001	0.061±0.001	0.033±0.001
LIFTed version	*One-error* ↓								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.409±0.007	0.411±0.009	0.397±0.003	0.644±0.003	0.643±0.004	0.379±0.006	0.561±0.010	0.197±0.004	0.131±0.003
LIFTed ECC	0.435±0.009	0.428±0.008	0.522±0.005	0.674±0.006	0.680±0.005	0.516±0.014	0.661±0.007	0.219±0.004	0.138±0.003
LIFTed version	*Coverage* ↓								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.095±0.001	0.100±0.003	0.114±0.003	0.266±0.002	0.256±0.002	0.116±0.003	0.113±0.002	0.117±0.001	0.159±0.001
LIFTed ECC	0.266±0.004	0.298±0.010	0.465±0.005	0.605±0.004	0.608±0.004	0.534±0.010	0.433±0.006	0.227±0.003	0.464±0.004
LIFTed version	*Ranking loss* ↓								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.036±0.001	0.039±0.001	0.063±0.002	0.132±0.002	0.127±0.001	0.068±0.002	0.097±0.002	0.048±0.001	0.046±0.001
LIFTed ECC	0.138±0.004	0.163±0.007	0.316±0.004	0.336±0.003	0.341±0.004	0.411±0.008	0.425±0.007	0.108±0.002	0.177±0.002
LIFTed version	*Average precision* ↑								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.636±0.003	0.648±0.004	0.563±0.004	0.350±0.002	0.343±0.002	0.586±0.004	0.482±0.007	0.820±0.002	0.717±0.002
LIFTed ECC	0.571±0.007	0.575±0.005	0.375±0.005	0.269±0.002	0.255±0.003	0.397±0.009	0.335±0.007	0.776±0.002	0.604±0.004
LIFTed version	*Macro-averaging AUC* ↑								
	rcv1-s1	rcv1-s2	bibtex	corel16k-s1	corel16k-s2	eurlex-sm	eurlex-dc	tmc2007	mediamill
LIFTed CLR	0.924±0.003	0.907±0.006	0.916±0.002	0.744±0.003	0.756±0.002	0.839±0.004	0.835±0.006	0.910±0.001	0.774±0.003
LIFTed ECC	0.685±0.006	0.664±0.006	0.618±0.003	0.533±0.001	0.532±0.002	0.552±0.001	0.528±0.001	0.784±0.003	0.525±0.001

incorporated into ECC and RAKEL to deal with the inherent randomness in their learning procedure, similar strategy may also be utilized by LIFT to account for the randomness in its clustering procedure (Table 1, step 3).

3) Furthermore, Fig. 1 shows that LIFT achieves comparable performance against CLR in terms of each evaluation metric. However, CLR needs to train $\binom{q}{2}$ binary classifiers leading to *quadratic* (i.e. $O(q^2)$) computational complexity. On the other hand, LIFT owns preferable scalability with only *linear* (i.e. $O(q)$) computational complexity.

4) As shown in Tables 3 and 4, across all evaluation metrics, LIFT ranks *1st* in 50.0% cases on the text data sets (*language log*, *enron*, *slashdot*, *rcv1-s1*, *rcv1-s2*, *bibtex*, *eurlex-sm*, *eurlex-dc* and *tmc2007*). On the other hand, LIFT ranks *1st* in more than 65.0% cases on the images data sets (*image*, *scene*, *corel5k*, *corel16k-s1* and *corel16k-s2*). These results indicate that LIFT tends to work better in application domains with dense feature representation (e.g. images) than those with sparse feature representation

TABLE 7
Wilcoxon signed-ranks test for BR, CLR and ECC against their LIFTed versions in terms of each evaluation metric (significance level $\alpha = 0.05$; p-values shown in the brackets).

Comparing algorithm	*Hamming loss*	*One-error*	*Coverage*	*Ranking loss*	*Avg. precision*	*Macro-avg. AUC*
LIFTed BR **versus** BR	**win** [7.7e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]
LIFTed CLR **versus** CLR	**win** [3.2e-3]	**win** [4.4e-2]	**tie** [6.0e-1]	**tie** [6.0e-1]	**tie** [4.4e-1]	**tie** [6.9e-1]
LIFTed ECC **versus** ECC	**tie** [1.3e-1]	**tie** [8.1e-1]	**loss** [1.2e-3]	**loss** [1.9e-3]	**loss** [1.1e-2]	**loss** [2.1e-3]

(e.g. text).

5) Furthermore, across all evaluation metrics, LIFT ranks *1st* in 60.4% cases on regular-scale data sets (Table 3). On the other hand, LIFT ranks *1st* in 46.3% cases on large-scale data sets (Table 4). These results indicate that LIFT tends to work better with moderate number of training examples.

To summarize, LIFT achieves highly competitive performance against other well-established multi-label learning algorithms, and the performance advantage is more pronounced on data sets with regular-scale and dense feature presentation.

4.3 Generality of Label-Specific Features

As shown in the above subsection, LIFT can significantly improve the performance of BR by employing label-specific features in inducing each binary classifier. Specifically, the label-specific features generation process for LIFT (Table 1, steps 2 to 4) is applicable to binary classification problem with positive training instances $\mathcal{P}$ and negative training instances $\mathcal{N}$. Therefore, label-specific features can be applied *internally* to any multi-label learning algorithm whenever its learning system comprises a number of binary classifiers. In this subsection, the generality of label-specific features is studied by incorporating them into two state-of-the-art multi-label learning algorithms CLR [14] and ECC [38].

CLR works by constructing a total of $\binom{q}{2}$ binary classifiers, each for a pair of possible class labels. Given a label pair (l_j, l_k) $(1 \leq j < k \leq q)$, the corresponding positive and negative training instances are determined as $\mathcal{P} = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_j \in Y_i, l_k \notin Y_i\}$ and $\mathcal{N} = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_j \notin Y_i, l_k \in Y_i\}$. By applying the label-specific generation process of LIFT on $\mathcal{P}$ and $\mathcal{N}$ while keeping other components of CLR unchanged, a LIFTed version of CLR can be instantiated.

ECC works by constructing a chain of q binary classifiers, where each binary classifier in the chain takes predictions of preceding ones as extra input features. To induce the binary classifier in the chain for class label l_k, the corresponding positive and negative training instances are determined as $\mathcal{P} = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_k \in Y_i\}$ and $\mathcal{N} = \{\boldsymbol{x}_i \mid (\boldsymbol{x}_i, Y_i) \in \mathcal{D}, l_k \notin Y_i\}$. By applying the label-specific generation process of LIFT on $\mathcal{P}$ and $\mathcal{N}$ while keeping other components of ECC unchanged, a LIFTed version of ECC can be instantiated.

Table 6 reports the detailed experimental results of LIFTed CLR and LIFTed ECC on all the experimental data sets.[8] Similar to Subsection 4.2, linear kernel LIBSVM [6] is employed as the binary learner for LIFTed CLR and LIFTed ECC, and the ensemble size for LIFTed ECC is set to be 100.

For CLR (Tables 3 and 4) and its LIFTed version (Table 6), out of the 17 benchmark data sets, LIFTed CLR achieves better performance on 12, 13, 12, 11, 9 and 12 of them in terms of *hamming loss*, *one-error*, *coverage*, *ranking loss*, *average precision* and *macro-averaging AUC* respectively. Accordingly, for ECC (Tables 3 and 4) and its LIFTed version (Table 6), LIFTed ECC only achieves better performance on 10, 9, 3, 3, 4 and 3 data sets in terms of each evaluation metric respectively.

Furthermore, to show whether the LIFTed version performs significantly better than the original algorithm, the traditional Wilcoxon signed-ranks test [10], [50] is employed to serve this purpose. Table 7 summarizes the statistical test results at significance level $\alpha = 0.05$, where the p-values for the corresponding tests are also shown in the brackets.

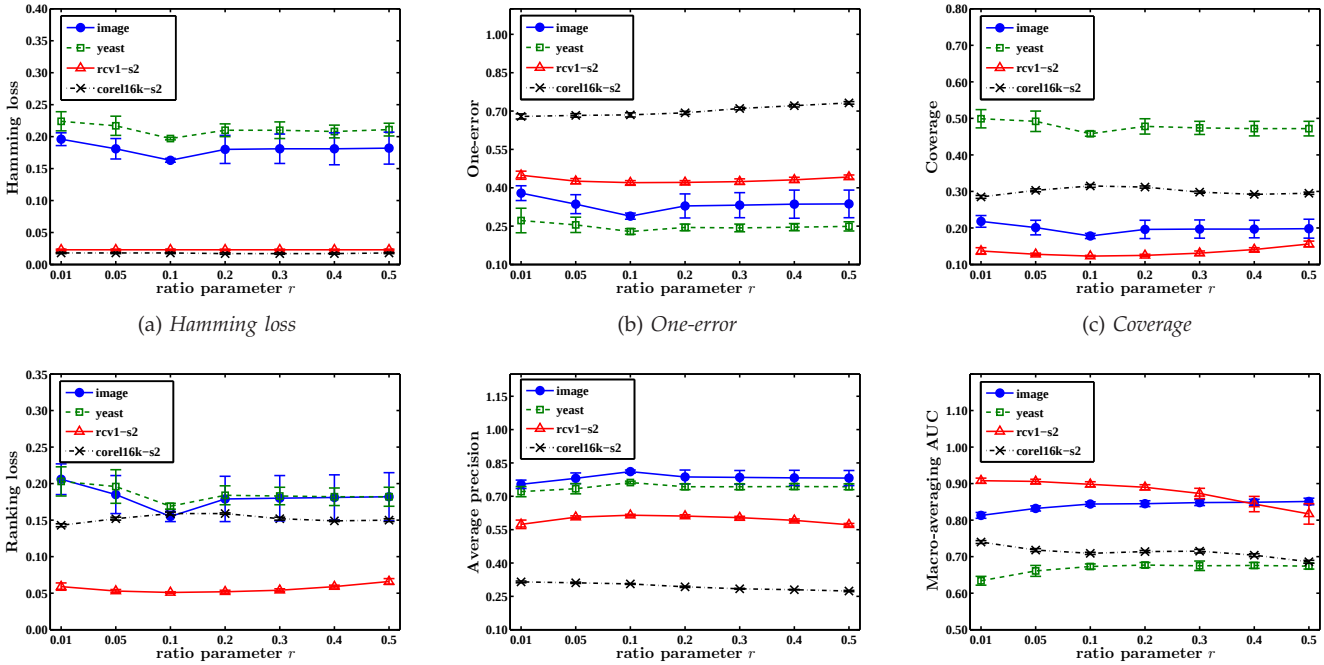
As shown in Table 7, label-specific features do help improve performance of BR and CLR, where the LIFTed versions achieve statistically superior or at least comparable performance in terms of each evaluation metric. On the other hand, LIFTed ECC performs inferiorly to ECC in terms of *coverage*, *ranking loss*, *average precision* and *macro-averaging AUC*. One possible reason might be that the extra features introduced by preceding classifiers in the chain are not well compatible with the label-specific features created in the k-means clustering procedure (Table 1, step 3). In summary, these results indicate that LIFT's label-specific features are capable of providing more useful representation than the original features, while its generality in improving multi-label learning algorithms' performance needs to be further investigated.

4.4 Auxiliary Results

4.4.1 Properties of LIFT

In this subsection, the parameter sensitivity as well as efficiency of LIFT will be further studied. To this end, additional experiments are conducted on two regular-scale data sets *image*, *yeast* and two large-scale data sets *rcv1-s2*, *corel16k-s2* (with the same experimental setup as

8. In this sense, the LIFT algorithm studied in Subsection 4.2 can be regarded as LIFTed BR.



(a) *Hamming loss* (b) *One-error* (c) *Coverage*

(d) *Ranking loss* (e) *Average precision* (f) *Macro-averaging AUC*

Fig. 2. Performance of LIFT changes in terms of each evaluation metric (mean±std. deviation) as the ratio parameter r increases from 0.01 to 0.5 on four benchmark multi-label data sets.

TABLE 8
Execution time of each comparing algorithm (mean±std. deviation) on four benchmark multi-label data sets.

Execution time	Data set	Comparing algorithm				
		LIFT	BR	CLR	ECC	RAKEL
Training phase (in minutes)	image	0.101±0.002	0.057±0.002	0.083±0.005	5.314±0.526	0.245±0.011
	yeast	0.286±0.010	0.097±0.004	0.243±0.012	9.994±0.133	0.450±0.012
	rcv1-s2	24.378±1.186	43.982±1.346	67.423±2.905	4952.000±122.827	246.675±27.620
	corel16k-s2	130.258±3.435	195.257±8.187	377.542±14.047	27650.351±379.670	1870.362±41.279
Testing phase (in seconds)	image	2.058±0.071	0.719±0.026	0.778±0.012	22.951±0.422	1.052±0.333
	yeast	5.469±0.431	0.945±0.034	1.149±0.044	42.887±0.906	0.992±0.016
	rcv1-s2	1174.895±114.488	22.568±0.975	1132.485±6.224	4155.692±61.199	73.078±1.720
	corel16k-s2	3207.250±308.109	115.006±3.887	9280.370±236.442	11669.953±298.015	333.022±5.548

in Subsection 4.2). Due to page limit, although experimental results on only four data sets are reported, similar observations hold for the other data sets as well.

• *Parameter Sensitivity*: As shown in Table 1, the only parameter needed to be specified for LIFT is r, which is set to be 0.1 in Subsection 4.2. To study the sensitivity of LIFT with respect to r, Fig. 2 illustrates how the performance of LIFT changes with increasing value of r in terms of each evaluation metric. Specifically, the ratio parameter r successively takes values of 0.01, 0.05, 0.1, 0.2, 0.3, 0.4 and 0.5. It is obvious from Fig. 2 that in most cases: (a) LIFT has slightly worse performance when the value of r is small ($r = 0.01$ and $r = 0.05$); b) The performance of LIFT becomes stable as the value of r increases beyond 0.1. Therefore, these observations justify LIFT's parameter setting in Subsection 4.2.

• *Execution Time*: To study the runtime efficiency of LIFT, Table 8 records the execution time (training phase as well as testing phase) of all the comparing algorithms investigated in Subsection 4.2. Each comparing algorithm is implemented within the MULAN multi-label learning library [44], which is built upon the well-known WEKA platform [19]. A Linux server equipped with Intel Xeon CPU (48 cores @ 2.67GHz) and 250GB memory is used for supporting the experiments. As shown in Table 8, on regular-scale data sets (*image*, *yeast*), LIFT consumes comparable execution time to the other algorithms. On large-scale data sets (*rcv1-s2*, *corel16k-s2*), LIFT consumes least execution time in training phase and is comparable to CLR and more efficient than ECC in the testing phase. These results validate the efficiency of LIFT in learning from multi-label data.

TABLE 9
Wilcoxon signed-ranks test for LIFT against LIFT-IG, LIFT-MLF and MLLS in terms of each evaluation metric (significance level $\alpha = 0.05$; p-values shown in the brackets).

Comparing algorithm	*Hamming loss*	*One-error*	*Coverage*	*Ranking loss*	*Avg. precision*	*Macro-avg. AUC*
LIFT **versus** LIFT-IG	**win** [1.2e-4]	**win** [2.9e-4]	**win** [4.2e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [3.5e-4]
LIFT **versus** LIFT-MLF	**win** [6.1e-5]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]	**win** [2.9e-4]
LIFT **versus** MLLS	**win** [4.9e-4]	**tie** [2.5e-1]	**tie** [8.8e-2]	**win** [4.4e-2]	**tie** [8.8e-1]	**win** [1.0e-3]

4.4.2 Complementary Comparing Algorithms

In a broader sense, generating label-specific features (i.e. creating the mapping function ϕ_k in Eq.(3)) could be viewed as a form of feature manipulation mechanism to multi-label learning. By retaining the *classification models induction* process of LIFT (Table 1, steps 6 to 9), variants of LIFT can be instantiated by replacing the *label-specific features construction* process (Table 1, steps 1 to 5) with other feature manipulation mechanisms:

• *Feature selection*: One straightforward feature manipulation mechanism to multi-label learning is to conduct supervised feature selection on each class label. Here, a subset of features $\mathcal{X}^{(k)} = \mathbb{R}^{d_k}$ are selected by keeping those with highest *information gain* in discriminating l_k. Let $\boldsymbol{x}^{(k)}$ be the feature vector derived by projecting $\boldsymbol{x}$ on $\mathcal{X}^{(k)}$. Thereafter, the mapping functions ϕ_k $(1 \leq k \leq q)$ are set by associating each selected feature subset to one class label, i.e.:

$$\phi_k(\boldsymbol{x}) = \boldsymbol{x}^{(k)} \tag{7}$$

The resulting variant of LIFT is termed as LIFT-IG.

• *Meta-level features*: Another feature manipulation mechanism to multi-label learning is to derive meta-level features from the original feature space to capture relationships between instances and class labels. Here, meta-level features employed by the MLF method [52] are used to instantiate the third variant of LIFT. MLF transforms each instance $\boldsymbol{x}$ into a meta-level feature vector $[\psi(\boldsymbol{x}, l_1), \psi(\boldsymbol{x}, l_2), \cdots, \psi(\boldsymbol{x}, l_q)]$ with $q \cdot (3r + 2)$ features, where $\psi(\boldsymbol{x}, l_k)$ $(1 \leq k \leq q)$ is a $(3r + 2)$-dimensional meta-level representation derived from the r nearest neighbors of $\boldsymbol{x}$ in $\mathcal{P}_k$. Specifically, $\psi(\boldsymbol{x}, l_k)$ corresponds to the concatenation of the L_2-distance, L_1-distance, and cosine similarity distance of $\boldsymbol{x}$ with respect to each of the r nearest neighbors, as well as the L_2-distance and cosine similarity distance of $\boldsymbol{x}$ with respect to the centroid of $\mathcal{P}_k$. Thereafter, the mapping functions ϕ_k are set by associating each meta-level representation $\psi(\boldsymbol{x}, \cdot)$ to one class label, i.e.:

$$\phi_k(\boldsymbol{x}) = \psi(\boldsymbol{x}, l_k) \tag{8}$$

The resulting variant of LIFT is termed as LIFT-MLF.

In addition to the above variants of LIFT, another work closely-related to label-specific features is also studied in this subsection:

• *Shared subspace*: In shared subspace model [23], [24], the learning system is composed of q linear classifiers defined over an extended instance space with shared features:

$$f_k(\boldsymbol{x}) = (\boldsymbol{w}_k + \boldsymbol{\Theta}^\top \boldsymbol{v}_k)^\top \boldsymbol{x} \quad (1 \leq k \leq q) \tag{9}$$

Here, $\boldsymbol{\Theta}$ is an $r \times d$ linear transformation matrix with orthonormal rows ($\boldsymbol{\Theta}\boldsymbol{\Theta}^\top = \boldsymbol{I}$) parameterizing the r-dimensional shared subspace ($r \ll d$). Therefore, $\boldsymbol{\Theta} \cdot \boldsymbol{x}$ can be regarded as a common part shared by all class labels which is handled by weight vector $\boldsymbol{v}_k$ specific to each class label. The model parameters $\boldsymbol{\Theta}$, $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_q]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_q]$ are optimized by solving a regularized least squares problem, where the corresponding algorithm is termed as MLLS.

For LIFT-IG, the number of selected features (i.e. d_k) is set to be the same as the number of features constructed by LIFT (i.e. $2m_k$). For LIFT-MLF, the number of nearest neighbors (i.e. r) are tuned within the range [10,100] as suggested in the literature. As a result, r is set to be 10 which turns out to yield stable performance. Similar to LIFT, LIBSVM (with linear kernel) is also employed as the binary learner for classifier induction for LIFT-IG and LIFT-MLF. For MLLS, the dimensionality of shared subspace (i.e. r) is set to be $q - 1$ as suggested in the literature.

Following the same experimental setup as used in Subsection 4.2, experiments on LIFT-IG, LIFT-MLF and MLLS are conducted on the regular-scale as well as large-scale data sets. Furthermore, Wilcoxon signed-ranks test [10], [50] is employed to show whether LIFT performs significantly better than LIFT-IG, LIFT-MLF and MLLS. Table 9 summarizes the statistical test results at significance level $\alpha = 0.05$, where the p-values for the corresponding tests are also shown in the brackets.[9]

As shown Table 9, LIFT achieves statistically superior performance than LIFT-IG and LIFT-MLF in terms of each evaluation metric. In addition, LIFT outperforms MLLS in terms of *hamming loss*, *ranking loss* and *macro-averaging AUC* and is comparable to MLLS in terms of the other evaluation metrics. These results clearly validate the effectiveness of LIFT's label-specific features generation process based on clustering analysis.

5 CONCLUSION

There have been numerous multi-label algorithms which learn from training examples by manipulating the label

9. Detailed experimental results on LIFT-IG, LIFT-MLF and MLLS can be found in the online complementary file.

space, i.e. exploiting label correlations. In this paper, an extension to our preliminary research [55] is presented which learns from training examples by manipulating the input space, i.e. exploiting features specific to each class label. The major contribution of our work is to utilize label-specific features for multi-label learning, which suggests a promising direction for learning from multi-label data.[10]

Experiments across the largest number of benchmark data sets up to date show that: (a) LIFT achieves highly competitive performance against other state-of-the-art multi-label learning algorithms; (b) Multi-label learning algorithms comprising binary classifiers might be improved by utilizing label-specific features; (c) Exploiting label-specific features is rather effective compared to other feature manipulation mechanisms. In the future, it is interesting to design other label-specific features generation strategies, incorporate label-specific features into other multi-label learning algorithms such as ML-KNN [58], and investigate how to consider label correlations in generating label-specific features.

REFERENCES

[1] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

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

[3] 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.

[4] R. S. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino, "Matrix completion for multi-label image classification," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds. Cambridge, MA: MIT Press, 2011, pp. 190–198.

[5] N. Cesa-Bianchi, M. Re, and G. Valentini, "Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference," *Machine Learning*, vol. 88, no. 1-2, pp. 209–241, 2012.

[6] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. Article 27, 2011.

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

[8] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Lecture Notes in Computer Science 2168*, L. De Raedt and A. Siebes, Eds. Berlin: Springer, 2001, pp. 42–53.

[9] F. D. Comité, R. Gilleron, and M. Tommasi, "Learning multi-label altenating decision tree from texts and data," in *Lecture Notes in Computer Science 2734*, P. Perner and A. Rosenfeld, Eds. Berlin: Springer, 2003, pp. 35–49.

[10] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.

[11] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188–230, 2004.

[12] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[13] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, pp. 681–687.

10. Source code of LIFT (with usage guide) is publicly-available at http://cse.seu.edu.cn/PersonalPage/zhangml/files/LIFT.rar.

[14] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.

[15] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, Bremen, Germany, 2005, pp. 195–200.

[16] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Lecture Notes in Artificial Intelligence 3056*, H. Dai, R. Srikant, and C. Zhang, Eds. Berlin: Springer, 2004, pp. 22–30.

[17] K. Gold and A. Petrosino, "Using information gain to build meaningful decision forests for multilabel classification," in *Proceedings of the 9th IEEE International Conference on Development and Learning*, Ann Arbor, MI, 2010, pp. 58–63.

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

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[20] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[21] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," *Artificial Intelligence*, vol. 172, no. 16, pp. 1897–1916, 2008.

[22] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[23] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proceedings of the 14th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, 2008, pp. 381–389.

[24] S. Ji, L. Tang, S. Yu, and J. Ye, "A shared-subspace learning framework for multi-label classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 2, 2010, Article 8.

[25] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proceedings of the ECML PKDD 2008 Discovery Challenge*, Antwerp, Belgium, 2008, pp. 75–83.

[26] A. Kumar, S. Vembu, A. K. Menon, and C. Elkan, "Learning and inference in probabilisic classifier chains with beam search," in *Lecture Notes in Computer Science 7523*, P. A. Flach, T. D. Bie, and N. Cristianini, Eds. Berlin: Springer, 2012, pp. 665–680.

[27] H.-Y. Lo, J.-C. Wang, H.-M. Wang, and S.-D. Lin, "Cost-sensitive multi-label learning for audio tag annotation and retrieval," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 518–529, 2011.

[28] E. Loza Mencía and J. Fürnkranz, "Pairwise learning of multilabel classifications with perceptrons," in *Proceedings of the International Joint Conference on Neural Networks*, Hong Kong, 2008, pp. 2899–2906.

[29] G. Madjarov, D. Gjorgjevikj, and T. Delev, "Efficient two stage voting architecture for pairwise multi-label classification," in *Lecture Notes in Computer Science 6464*, J. Li, Ed. Berlin: Springer, 2011, pp. 164–173.

[30] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, 2012.

[31] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *Working Notes of the AAAI'99 Workshop on Text Learning*, Orlando, FL, 1999.

[32] E. Montañes, R. Senge, J. Barranquero, J. Ramón Quevedo, J. José del Coz, and E. Hüllermeier, "Dependent binary relevance models for multi-label classification," *Pattern Recognition*, vol. 47, no. 3, pp. 1494–1508, 2014.

[33] K. Ozonat and D. Young, "Towards a universal marketplace over the web: Statistical multi-label classification of service provider forms with simulated annealing," in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Paris, France, 2009, pp. 1295–1303.

[34] F. Pachet and P. Roy, "Improving multilabel analysis of music titles: A large-scale validation of the correction approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 2, pp. 335–343, 2009.

[35] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *Proceedings of the 15th*

ACM International Conference on Multimedia, Augsburg, Germany, 2007, pp. 17–26.

[36] R. Rak, L. Kurgan, and M. Reformat, "Multi-label associative classification of medical documents from medline," in *Proceedings of the 4th International Conference on Machine Learning and Applications*, Los Angeles, CA, 2005, pp. 177–186.

[37] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proceeding of the 8th IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 995–1000.

[38] 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.

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

[40] R. E. Schapire and Y. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[41] C. G. M. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. M. Smeulders, "The challenge problem for automated detection of 101 semantic concepts in multimedia," in *Proceedings of the 14th Annual ACM International Conference on Multimedia*, Santa Barbara, CA, USA, 2006, pp. 421–430.

[42] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proceedings of the 9th International Conference on Music Information Retrieval*, Philadephia, PA, 2008, pp. 325–330.

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

[44] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411–2414, 2011.

[45] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, "Tutorial on learning from multi-label data," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Bled, Slovenia, 2009 [http://www.ecml pkdd2009.net/wp-content/uploads/2009/08/learning-from-multi-label-data.pdf].

[46] N. Ueda and K. Saito, "Parametric mixture models for multi-label text," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 721–728.

[47] H. Wang, C. Ding, and H. Huang, "Multi-label classification: Inconsistency and class balanced k-nearest neighbor," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, Atlanta, GA, 2010, pp. 1264–1266.

[48] J. Wang, Y. Zhao, X. Wu, and X.-S. Hua, "A transductive multi-label learning approach for video concept detection," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2274–2286, 2011.

[49] M. Wang, X. Zhou, and T.-S. Chua, "Automatic image annotation via local multi-label classification," in *Proceedings of the 7th ACM International Conference on Image and Video Retrieval*, Niagara Falls, Canada, 2008, pp. 17–26.

[50] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[51] R. Yan, J. Tešić, and J. R. Smith, "Model-shared subspace boosting for multi-label classification," in *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, San Jose, CA, 2007, pp. 834–843.

[52] Y. Yang and S. Gopal, "Multilabel classification with meta-level features in a learning-to-rank framework," *Machine Learning*, vol. 88, no. 1-2, pp. 47–68, 2012.

[53] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, 1997, pp. 412–420.

[54] K. Yu, S. Yu, and V. Tresp, "Multi-label informed latent semantic indexing," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil, 2005, pp. 258–265.

[55] M.-L. Zhang, "LIFT: Multi-label learning with label-specific features," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 1609–1614.

[56] M.-L. 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 D. C., 2010, pp. 999–1007.

[57] M.-L. Zhang and Z.-H. 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, 2006.

[58] M.-L. Zhang and Z.-H. Zhou, "ML-kNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[59] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, in press.

[60] S. Zhu, X. Ji, W. Xu, and Y. Gong, "Multi-labelled classification using maximum entropy method," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil, 2005, pp. 274–281.