This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering

Learning Label-Specific Features and Class-Dependent Labels for Multi-Label Classification

Jun Huang, *Student Member, IEEE,* Guorong Li, *Member, IEEE,* Qingming Huang, *Senior Member, IEEE,* and Xindong Wu, *Fellow, IEEE*

Abstract—Binary Relevance is a well-known framework for multi-label classification, which considers each class label as a binary classification problem. Many existing multi-label algorithms are constructed within this framework and utilize identical data representation in the discrimination of all the class labels. In multi-label classification, however, each class label might be determined by some specific characteristics of its own. In this paper, we seek to learn *label-specific data representation* for each class label, which is composed of *label-specific features*. Our proposed method LLSF can not only be utilized for multi-label classification directly, but also be applied as a feature selection method for multi-label learning and a general strategy to improve multi-label classification algorithms comprising a number of binary classifiers. Inspired by the research works on modeling *high-order* label correlations, we further extend LLSF to learn *class-Dependent Labels* in a *sparse stacking* way, denoted as LLSF-DL. It incorporates both *second-order* and *high-order* label correlations. A comparative study with the state-of-the-art approaches manifests the effectiveness and efficiency of our proposed methods.

Keywords—Multi-Label Classification, Label Correlation, Feature Selection.

1 Introduction

Multi-label learning deals with examples having multiple class labels simultaneously. It has attracted significant attention from researchers and has been applied to a variety of domains, such as text categorization [1], [2], image annotation [3], [4], video annotation [5], [6], social networks [7], music emotion categorization [8], [9]. The challenge is how to learn a well-constructed classification model which can predict a set of possible labels for unseen examples. The simplest approach for multi-label classification is to perform *problem transformation* [10], [11], where a multi-label problem is transformed into one or more single-label subproblems. BR [3], one of the representative algorithms of *problem transformation*, considers each class label as an independent binary (one-vs-rest) classification problem. Then, traditional single-label classification approaches, e.g., Logistic Regression, Naive Bayesian, and SVM, can be used to solve these binary classification problems directly.

The BR approach is theoretically simple and intuitive, and many existing advanced multi-label classification algorithms are built within this framework, such as [12], [13], [14], [15], [16], [17], [18], [19]. In all the binary classification subproblems, the output space is different, but the input space is the same. It implies that the single data representation which is represented by all the features of a data set is utilized in the discrimination of all the class labels. In multi-label learning, however, the examples belong to multiple labels simultaneously, and each class label might be determined by some specific features of its own and these features are the most *pertinent* and *discriminative* features to the corresponding class label. For example, features like *height* and *weight* have strong discriminability in the discrimination of whether a person is a basketball player. Similarly, features *coding* and *education* have strong discriminability when determining whether a person is a computer engineer. These features could be considered as *label-specific features* to the corresponding class label.

Traditional feature selection methods for single-label classification can be employed to obtain *label-specific features* for multi-label classification, e.g., Information Gain, Laplacian Score, and Fisher Score. However, label correlation among class labels is neglected. Some feature selection, dimension reduction and subspace learning methods have been proposed to learn discriminative features for multi-label classification, e.g., S-CLS[20], MDDM [13], and MLLS [12], but the low dimensional data representation learned by these algorithms is shared by all the class labels. LIFT [19] utilizes *label-specific features* to represent instances for predicting the corresponding class label. It can be viewed as a feature mapping method, and lacks of interpretability, i.e. it is not clear that which features have discriminability to each class label. Besides, it does not take label correlation information into consideration. Meanwhile, several works in multi-task learning [21], [22] have been proposed to learn common-across-tasks features and task-specific features, which yield good results.

Inspired by previous works, we first proposed **L**earning **L**abel-**S**pecific **F**eatures for each class label by considering *pairwise* (i.e. *second-order*) label correlations in our preliminary version [23], denoted as **LLSF**. We assume that the *label-specific features* have the following three properties:

1) ***discriminability***: The *label-specific features* should be the most *pertinent* and *discriminative* features to the corresponding class label.
2) ***sparsity***: Each class label is only determined by a *subset* of relevant features of a given data set.
3) ***sharing***: Any two strongly correlated class labels can *share* more features with each other than two uncorrelated or weakly correlated ones.

Fig. 1 shows the learning structure of LLSF. It exploits *label-specific features* for the discrimination of each class label, and can be directly applied to multi-label classification. Besides, it can be utilized as a feature selection method for multi-label learning and a general strategy to improve multi-label classification algorithms comprising a number of binary classifiers. LLSF only exploits the pairwise label correlation for multi-label classification. Meanwhile,

- *J. Huang and G. Li are with the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 101480, China. E-mail: huangjun13b@mails.ucas.ac.cn, liguorong@ucas.ac.cn.*
- *Q. Huang is with the School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 101480, China, and the Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China E-mail: qmhuang@ucas.ac.cn.*
- *X. Wu is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China, and the Department of Computer Science, University of Vermont, 33 Colchester Avenue, Burlington, VT 05405. E-mail: xwu@cs.uvm.edu.*

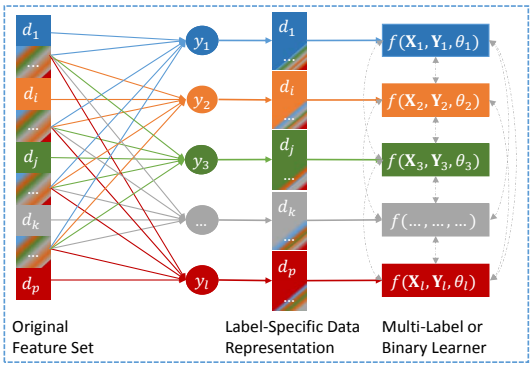


Fig. 1. Label-specific features for each label are indicated in different colors, and dashed arrows represent the output of one binary classifier can be used as input for the others, e.g., in Classifier Chain (CC).

a high-order strategy has stronger correlation-modeling capabilities than second-order strategies. We expect that LLSF could obtain more competitive performance if we extend it to model high-order label correlation. Modeling *high-order* label correlations has won a great success in multi-label classification [24], [25], [26], [27], [28], [29], [30], [31], but many of the previous works fail to pay adequate attention to two critical issues [32]: error propagation and unnecessary dependent relationships.

To overcome these drawbacks, we further extend **LLSF** to model *high-order* label correlations by learning class-**D**ependent **L**abels in a *sparse stacking* way, denoted as **LLSF-DL**. Our major contributions are summarized as follows:

- We propose to learn *label-specific data representation* for multi-label classification by considering label correlations.
- We further extend LLSF to learn *class-dependent labels*, which exploits both *second-order* and *high-order* label correlations.
- The proposed methods LLSF and LLSF-DL can not only be utilized for multi-label classification directly, but also be applied as a feature selection method for multi-label learning and a general strategy to improve multi-label classification algorithms comprising a number of binary classifiers.
- The experimental results on fifteen multi-label data sets show the effectiveness and efficiency of our methods against the state-of-the-art multi-label classification algorithms.

Difference to conference version. Compared to our conference version paper [23], the extended work in this paper mainly includes: 1). We extend LLSF to exploit *high-order* label correlations by learning class-Dependent Labels in a *sparse stacking* way. 2). We significantly strengthen the experiment part: (a) fifteen data sets and nine comparing algorithms are used to evaluate the effectiveness of our proposed methods. (b) the performance of feature selection of LLSF is evaluated. (c) statistical tests, e.g., Friedman test, Nemenyi test and Wilcoxon signed-ranks test, are added to test whether our proposed methods achieve a competitive performance against the comparing algorithms.

The rest of this paper is organized as follows. Section 2 reviews previous works on multi-label learning. Section 3 presents details of the proposed methods LLSF and LLSF-DL. Experimental results and analyses are shown in Section 4. Section 5 shows the results of LLSF and LLSF-DL on large-scale multi-label data. Finally, we conclude the paper in Section 6.

2 Related Work

Research on multi-label learning originates from text categorization [1]. Over the past decades, many well-established algorithms have been proposed to solve multi-label learning problems in various domains. According to the review papers [10], [11], these algorithms can be grouped into two categories: *problem transformation* (fitting data to algorithm) and *algorithm adaption* (fitting algorithm to data) approaches. Being a presentative framework of the first category, BR [3] is simple and straightforward. However, it suffers from several drawbacks: 1) BR does not take the correlation information among different labels into consideration. 2) it becomes computationally unaffordable for data sets with many labels. 3) the binary classifiers may suffer from the issue of class-imbalance. Many existing advanced multi-label classification algorithms are built within this framework to overcome different drawbacks or solve new problems in multi-label learning.

For the first problem, many algorithms have been proposed by mining *second-order* or *high-order* label correlations. Second-order approaches exploit *pairwise* relationships between labels, such as BP-MLL [2] and CLR [33]. High-order approaches tackle multi-label learning problem by mining relationships between all the class labels or subsets of class labels. CC [24] is a novel chain algorithm which models *high-order* label correlations by using the vector of class labels as additional features. It transforms a multi-label classification problem into a chain of l binary classification problems, and the i-th classifier h_i is trained by using the results of labels $y_1, y_2, ..., y_{i-1}$ as additional input information. To predict subsequent labels in a given chain order, CC resorts to using outputs of the preceding classifiers, which makes them prone to errors. The performance of CC is seriously constrained by the training order of labels and error propagation. Besides, it may not be appropriate that each label is dependent on all the preceding labels in a given chain order. PCC [34] is an extended work on CC by formulating a probabilistic interpretation. PCC suffers from the computational issue that the inference requires time exponential in the number of class labels. Moreover, the performance of PCC is sensitive to the order of class labels while training. There are several extended approaches on CC and PCC by searching for suitable order of labels or dependent structures among labels and reducing the computational complexity, e.g., HIROM [35], PruDent [32], BCC [25], LEAD [26], PCC-beam [28], and MCC [29]. Existing approaches mainly exploit label correlations globally by assuming that the label correlations are shared by all the examples. In real-world tasks, however, different examples may share different label correlations, and few correlations are globally applicable. There are several approaches on exploiting label correlations locally, such as ML-LOC [36] and GCC [37].

Learning from multi-label data with large number of labels, the number of requisite predictive models will be quite large, making the training costs unaffordable. To address this issue, researchers try to perform *label space dimension reduction* (LSDR), e.g., CS [14], PLST [15], CPLST [16], CL [17], and FaIE [18]. For LSDR, each original high-dimensional label vector is encoded to a low dimensional code vector in a latent space. Then predictive models are trained from the data matrix to code vectors, whose quantity is much smaller than the original and thus can significantly reduce the training costs. To predict an unseen example, a low-dimensional code vector is firstly obtained with the learned predictive models on its features and then efficiently decoded for recovering its predicted label vector. If the learned predictive models and the decoding process are effective enough, LSDR will be expected to yield acceptable classification performance at much lower costs. Class imbalance [38] is ubiquitous in Machine Learning, Data Mining and Pattern Recognition applications. Several approaches, e.g., MLSC [39], IRUS [40], and COCOA [41], have been proposed to tackle it in multi-label learning.

Existing feature selection, dimension reduction and subspace learning methods aim to learn a low dimensional data representa-

tion for multi-label classification, which is shared by all the class labels, e.g., gMLC [42], S-CLS[20], MDDM [13], and MLLS [12]. LIFT [19] exploits different feature sets to discriminate different class labels. The *label-specific features* are learned by conducting clustering analysis on the positive and negative examples for a given label, and represented by distances between the original instances and the instances in the centers of positive and negative examples. It can be viewed as a feature mapping method, and does not consider label correlations. Several approaches in multi-task learning, e.g., DirtyLasso [21], GFLasso [22], MixedNorm [43], and RMTL [44], have been proposed to learn common-across-tasks features and task-specific features, and yielded good results. DirtyLasso [21] utilizes $\ell_{1,\infty}$-norm and ℓ_1-norm to extract essential features shared by the tasks and task-specific features, respectively. DirtyLasso learns common features shared by all the tasks, and does not model relatedness between tasks explicitly. GFLasso [22] exploits a graph structure over the tasks and encourages highly correlated tasks to share a common set of relevant features by calculating distance between coefficient vectors of tasks, and ℓ_1-norm is employed to extract task-specific features.

Previous work mainly utilizes identical feature representation to discriminate all the class labels. In this paper, we propose to learn *label-specific features* and *class-dependent labels* for multi-label classification, which incorporates not only *second-order*, but also *high-order* label correlations.

3 Learning Label-Specific Features and Class-Dependent Labels

In this section, we first present how to model the properties of *label-specific features*, and then introduce how to exploit *second-order* and *high-order* label correlations. Last, we will give details of optimization of the proposed model via accelerated proximal gradient descent algorithm.

3.1 Notations

Let $\mathcal{X} = \mathbb{R}^p$ be an input space with p-dimensional and $\mathcal{Y} = \{y_1, y_2, ..., y_l\}$ be a finite set of l possible class labels. We denote the input data as a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^T \in \mathbb{R}^{n\times p}$, and the output labels as a matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]^T \in \{0,1\}^{n\times l}$. The i-th example is denoted by a vector with p attribute values $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{ip}]$, $\mathbf{x}_i \in \mathcal{X}$, and $\mathbf{y}_i = [y_{i1}, y_{i2}, ..., y_{il}]$ is a ground truth label of $\mathbf{x}_i$. Each element $y_{ij} = 1$ if the label y_j is associated with $\mathbf{x}_i$, otherwise $y_{ij} = 0$.

3.2 Discriminability and Sparsity of Label-Specific Features

Discriminability indicates that the label-specific features should be the most pertinent and discriminative to the corresponding class label, and *sparsity* indicates that each class label is only determined by a subset of relevant features of a given data set. We can generalize our algorithm as the following optimization problem

$$\min_{\mathbf{w}^i} \ell(f(\mathbf{X}\mathbf{w}^i), \mathbf{y}^i) + \beta\|\mathbf{w}^i\|_1 \tag{1}$$

where $\ell(\cdot)$ is a loss function, and it can be implemented with various loss functions. We choose the least square loss function because of its efficiency and simplicity reported in many applications. By applying the least square loss function, the optimization problem is then defined as

$$\min_{\mathbf{w}^i} \frac{1}{2}\|\mathbf{X}\mathbf{w}^i - \mathbf{y}^i\|_2^2 + \beta\|\mathbf{w}^i\|_1 \tag{2}$$

where $\mathbf{w}^i = [w_{1i}, w_{2i}, ..., w_{pi}]^T$ represents the regression parameter of the model for the i-th label, and $\mathbf{y}^i = [y_{1i}, y_{2i}, ..., y_{ni}]^T$ represents the i-th column of $\mathbf{Y}$, $1 \le i \le l$. Eq (2) equals Lasso [45]. The non-zero components w_{ji} of $\mathbf{w}^i$ indicate that the corresponding features are discriminative to label y_i. Consequently, these features could be considered as *label-specific features* of label y_i, and the number of them will be much smaller than p.

3.3 Incorporating Second-Order Label Correlations

In multi-label learning, class labels often have correlations with each other. For example, an object that belongs to label y_1 is also likely belongs to label y_2. Conversely, belonging to label y_1 can make an object less likely to belong to label y_3. If two class labels are strongly correlated, features discriminative to one class label might be discriminative to the other. On the contrary, if two class labels are uncorrelated or weakly correlated, the features discriminative to one class label might not be discriminative to the other. Therefore, we assume that two strongly correlated labels can share more features with each other than two uncorrelated or weakly correlated labels.

In LLSF, the *label-specific features* for the i-th label are determined by non-zero components of $\mathbf{w}^i$. If label y_i and label y_j are strongly correlated, features discriminative to y_i may be also discriminative to y_j with a higher probability. Then the two labels will share most of the *label-specific features* and the corresponding coefficients $\mathbf{w}^i$ and $\mathbf{w}^j$ will be very similar, and thus their inner product will be large; otherwise, the inner product will be small. We incorporate second-order (i.e. *pairwise*) label correlation by calculating the inner product between any pair of coefficient vectors to model the property of *sharing* of *label-specific features*, and the optimization problem is formulated as

$$\min_{\mathbf{w}^i} \frac{1}{2}\|\mathbf{X}\mathbf{w}^i - \mathbf{y}^i\|_2^2 + \frac{\alpha}{2}\sum_{j=1}^{l} r_{ij}\mathbf{w}^{iT}\mathbf{w}^j + \beta\|\mathbf{w}^i\|_1 \tag{3}$$

where $r_{ij} = 1 - c_{ij}$, c_{ij} represents the correlation between labels y_i and y_j, and $\mathbf{C} \in \mathbb{R}^{l\times l}$ is the label correlation matrix. In this paper, the correlation matrix $\mathbf{C}$ is calculated by cosine similarity.

Combing all the binary classifiers together, the final optimization objective of LLSF can be rewritten as

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \frac{\alpha}{2}\mathrm{Tr}(\mathbf{R}\mathbf{W}^T\mathbf{W}) + \beta\|\mathbf{W}\|_1 \tag{4}$$

where $\mathbf{W} = [\mathbf{w}^1, \mathbf{w}^2, ..., \mathbf{w}^l] \in \mathbb{R}^{p\times l}$, $\mathbf{R} \in \mathbb{R}^{l\times l}$ with elements r_{ij}, $\mathbf{Y} = [\mathbf{y}^1, \mathbf{y}^2, ..., \mathbf{y}^l] \in \mathbb{R}^{n\times l}$, $\alpha \ge 0$ and $\beta \ge 0$ are two tradeoff parameters.

3.4 Learning Class-Dependent Labels

In multi-label learning, each label might be dependent on more than one class label. According to [11], a *high-order* strategy has stronger correlation-modeling capabilities than *first-order* (i.e. do not model label correlations) and *second-order* strategies. But most works fail to pay adequate attention to two critical issues [32]: error propagation and unnecessary dependent relationships.

Inspired by previous works on modeling *high-order* label correlation by *stacking* [27], [32], we further extend LLSF to incorporate *high-order* label correlations in a *sparse stacking* way, denoted as LLSF-DL. A label space $\mathbf{Y}$ is augmented with the feature space $\mathbf{X}$ as additional features, and ℓ_1 norm regularization on $\mathbf{W}_y$ is employed to learn *sparse* label dependency relationships. The sparsity regularization on $\mathbf{W}_y$ could remove unnecessary dependency relationships. Thus, it can alleviate the negative influence of error propagations from unnecessary dependency relationships. The optimization of LLSF-DL is defined as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering

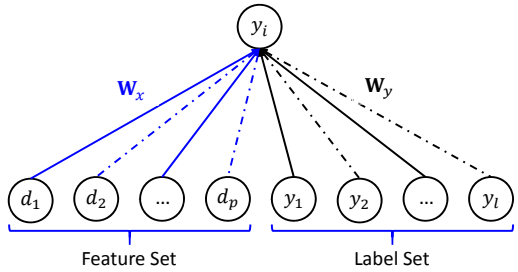


Fig. 2. LLSF-DL: Learning Label-Specific Features and class-Dependent Labels. The dashed arrows represent the corresponding features or labels are not specific to y_i.

$$\min_{\mathbf{W}_x,\mathbf{W}_y} \frac{1}{2}\|\mathbf{X}\mathbf{W}_x + \mathbf{Y}\mathbf{W}_y - \mathbf{Y}\|_F^2 + \frac{\alpha}{2}\mathrm{Tr}(\mathbf{R}\mathbf{W}_x^T\mathbf{W}_x) + \beta\|\mathbf{W}_x\|_1 + \gamma\|\mathbf{W}_y\|_1 \quad (5)$$

where $\mathbf{W}_x = [\mathbf{w}_x^1, \mathbf{w}_x^2, ..., \mathbf{w}_x^l] \in \mathbb{R}^{p\times l}$, $\mathbf{W}_y = [\mathbf{w}_y^1, \mathbf{w}_y^2, ..., \mathbf{w}_y^l] \in \mathbb{R}^{l\times l}$, $\alpha \geq 0$, $\beta \geq 0$ and $\gamma \geq 0$ are the tradeoff parameters, and the definition of $\mathbf{R}$ is the same as LLSF.

3.5 Accelerated Proximal Gradient

Although the minimization of (4) and (5) are two convex optimization problems, the objective functions are non-smooth due to the non-smoothness of the ℓ_1 norm regularization terms. We use the accelerated proximal gradient method to solve these non-smooth optimization problems.

A general accelerated proximal gradient method can be written as the following convex optimization problem

$$\min_{\mathbf{W}\in\mathcal{H}} \{F(\mathbf{W}) = f(\mathbf{W}) + g(\mathbf{W})\} \quad (6)$$

where $\mathcal{H}$ is a real Hilbert space, both $f(\mathbf{W})$ and $g(\mathbf{W})$ are convex. $f(\mathbf{W})$ is further *Lipschitz* continuous, i.e. $\|\nabla f(\mathbf{W}_1) - \nabla f(\mathbf{W}_2)\| \leq L_f\|\Delta\mathbf{W}\|$, where $\Delta\mathbf{W} = \mathbf{W}_1 - \mathbf{W}_2$, and L_f is the *Lipschitz* constant

Instead of directly minimizing $F(\mathbf{W})$, proximal gradient algorithms minimize a sequence of separable quadratic approximations to $F(\mathbf{W})$, denoted as

$$Q_{L_f}(\mathbf{W}, \mathbf{W}^{(t)}) = f(\mathbf{W}^{(t)}) + \langle\nabla f(\mathbf{W}^{(t)}), \mathbf{W} - \mathbf{W}^{(t)}\rangle + \frac{L_f}{2}\|\mathbf{W} - \mathbf{W}^{(t)}\|_F^2 + g(\mathbf{W}) \quad (7)$$

Let $\mathbf{G}^{(t)} = \mathbf{W}^{(t)} - \frac{1}{L_f}\nabla f(\mathbf{W}^{(t)})$, then

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} Q_{L_f}(\mathbf{W}, \mathbf{W}^{(t)}) = \arg\min_{\mathbf{W}} g(\mathbf{W}) + \frac{L_f}{2}\|\mathbf{W} - \mathbf{G}^{(t)}\|_F^2 \quad (8)$$

In [46], the work has shown that setting $\mathbf{W}^{(t)} = \mathbf{W}_t + \frac{b_{t-1}-1}{b_t}(\mathbf{W}_t - \mathbf{W}_{t-1})$ for a sequence b_t by satisfying $b_{t+1}^2 - b_{t+1} \leq b_t^2$ can improve the convergence rate to $O(t^{-2})$, where $\mathbf{W}_t$ is the result of $\mathbf{W}$ at the t-th iteration.

Before presenting the details of optimization for LLSF and LLSF-DL, we first introduce the soft-thresholding operator. For $w \in \mathbb{R}$ and $\varepsilon > 0$, the soft-thresholding operation is defined as

$$S_\varepsilon[w] = \begin{cases} w - \varepsilon & \text{if } w > \varepsilon \\ w + \varepsilon & \text{if } w < -\varepsilon \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

Algorithm 1: Optimization of LLSF

Input: Training data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, label matrix $\mathbf{Y} \in \mathbb{R}^{n\times l}$, and weighting parameters α, β, ρ.
Output: Coefficient matrix $\mathbf{W}^* \in \mathbb{R}^{p\times l}$.

Initialization:
$b_0, b_1 \leftarrow 1$; $t \leftarrow 1$; $\mathbf{W}_0, \mathbf{W}_1 \leftarrow (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$;
calculate the Lipschitz constant L_f by (23);
initialize $\mathbf{R}$ by calculating the cosine similarity on $\mathbf{Y}$;
repeat
  $\mathbf{W}^{(t)} \leftarrow \mathbf{W}_t + \frac{b_{t-1}-1}{b_t}(\mathbf{W}_t - \mathbf{W}_{t-1})$;
  $\mathbf{G}^{(t)} \leftarrow \mathbf{W}^{(t)} - \frac{1}{L_f}\nabla f(\mathbf{W}^{(t)})$;
  $\mathbf{W}_{t+1} \leftarrow \mathbf{S}_{\frac{\beta}{L_f}}(\mathbf{G}^{(t)})$;
  $b_{t+1} \leftarrow \frac{1+\sqrt{4b_t^2+1}}{2}$;
  $t \leftarrow t + 1$;
until *stop criterion reached*;
$\mathbf{W}^* \leftarrow \mathbf{W}_{t+1}$;

Proposition 3.1: If $\mathcal{H}$ is a Euclidean space endowed with the Frobenius norm $\|\cdot\|_F$ and $g(\cdot)$ is ℓ_1 norm, then $\mathbf{W}^{t+1}$ is given by soft-thresholding the entries of $\mathbf{G}^{(t)}$ as

$$\mathbf{W}^{t+1} = S_\varepsilon[\mathbf{G}^{(t)}] = \arg\min_{\mathbf{W}} \varepsilon\|\mathbf{W}\|_1 + \frac{1}{2}\|\mathbf{W} - \mathbf{G}^{(t)}\|_F^2 \quad (10)$$

3.6 Optimization of LLSF Model

According to (4) and (6), $f(\mathbf{W})$ and $g(\mathbf{W})$ are defined as follows

$$f(\mathbf{W}) = \frac{1}{2}\|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2 + \frac{\alpha}{2}\mathrm{Tr}(\mathbf{R}\mathbf{W}^T\mathbf{W}) \quad (11)$$

$$g(\mathbf{W}) = \beta\|\mathbf{W}\|_1 \quad (12)$$

According to (11), we can calculate $\nabla f(\mathbf{W})$ as

$$\nabla f(\mathbf{W}) = \mathbf{X}^T\mathbf{X}\mathbf{W} - \mathbf{X}^T\mathbf{Y} + \alpha\mathbf{W}\mathbf{R} \quad (13)$$

According to (8), (11) and (12), the coefficient matrix $\mathbf{W}$ can be optimized by

$$\begin{aligned}\mathbf{W}^* &= \arg\min_{\mathbf{W}} Q_{L_f}(\mathbf{W}, \mathbf{W}^{(t)}) \\ &= \arg\min_{\mathbf{W}} \tfrac{L_f}{2}\|\mathbf{W} - \mathbf{G}^{(t)}\|_F^2 + g(\mathbf{W}) \\ &= \arg\min_{\mathbf{W}} \tfrac{1}{2}\|\mathbf{W} - \mathbf{G}^{(t)}\|_F^2 + \tfrac{\beta}{L_f}\|\mathbf{W}\|_1\end{aligned} \quad (14)$$

where $\mathbf{G}^{(t)} = \mathbf{W}^{(t)} - \frac{1}{L_f}\nabla f(\mathbf{W}^{(t)})$.

Here $g(\mathbf{W})$ is ℓ_1 norm regularization. According to Proposition 3.1, in each iteration, $\mathbf{W}$ can be obtained by the following soft-thresholding operation,

$$\mathbf{W}^{t+1} = S_{\frac{\beta}{L_f}}[\mathbf{G}^{(t)}] \quad (15)$$

The overall procedures of optimization for LLSF via accelerated proximal gradient algorithm are summarized in Algorithm 1.

3.7 Optimization of LLSF-DL Model

According to (5) and (6), $f(\mathbf{W}_x, \mathbf{W}_y)$ and $g(\mathbf{W}_x, \mathbf{W}_y)$ can be defined as follows

$$f(\mathbf{W}_x, \mathbf{W}_y) = \frac{1}{2}\|\mathbf{X}\mathbf{W}_x + \mathbf{Y}\mathbf{W}_y - \mathbf{Y}\|_F^2 + \frac{\alpha}{2}\mathrm{Tr}(\mathbf{R}\mathbf{W}_x^T\mathbf{W}_x) \quad (16)$$

$$g(\mathbf{W}_x, \mathbf{W}_y) = \beta\|\mathbf{W}_x\|_1 + \gamma\|\mathbf{W}_y\|_1 \quad (17)$$

Algorithm 2: Optimization of LLSF-DL

Input: Training data matrix $\mathbf{X} \in \mathbf{R}^{n\times p}$, label matrix $\mathbf{Y} \in \mathbb{R}^{n\times l}$, and weighting parameters $\alpha, \beta, \gamma, \rho$.
Output: Coefficient matrix $\mathbf{W}_x^* \in \mathbb{R}^{p\times l}$ and $\mathbf{W}_y^* \in \mathbb{R}^{l\times l}$.

Initialization:
$b_0, b_1 \leftarrow 1$; $t \leftarrow 1$;;
$\mathbf{W}_x^0, \mathbf{W}_x^1 \leftarrow (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$;
$\mathbf{W}_y^0, \mathbf{W}_y^1 \leftarrow (\mathbf{Y}^T\mathbf{Y} + \rho\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{Y}$;
calculate the Lipschitz constant L_f by (24);
initialize $\mathbf{R}$ by calculating the cosine similarity on $\mathbf{Y}$;
repeat
  $\mathbf{W}_x^{(t)} \leftarrow \mathbf{W}_x^t + \frac{b_{t-1}-1}{b_t}(\mathbf{W}_x^t - \mathbf{W}_x^{t-1})$;
  $\mathbf{G}_x^{(t)} \leftarrow \mathbf{W}_x^{(t)} - \frac{1}{L_f}\nabla f(\mathbf{W}_x^{(t)})$;
  $\mathbf{W}_x^{t+1} \leftarrow S_{\frac{\beta}{L_f}}(\mathbf{G}_x^{(t)})$;
  $\mathbf{W}_y^{(t)} \leftarrow \mathbf{W}_y^t + \frac{b_{t-1}-1}{b_t}(\mathbf{W}_y^t - \mathbf{W}_y^{t-1})$;
  $\mathbf{G}_y^{(t)} \leftarrow \mathbf{W}_y^{(t)} - \frac{1}{L_f}\nabla f(\mathbf{W}_y^{(t)})$;
  $\mathbf{W}_y^{t+1} \leftarrow S_{\frac{\gamma}{L_f}}(\mathbf{G}_y^{(t)})$;
  $b_{t+1} \leftarrow \frac{1+\sqrt{4b_t^2+1}}{2}$;
  $t \leftarrow t+1$;
until *stop criterion reached*;
$\mathbf{W}_x^* \leftarrow \mathbf{W}_x^{t+1}$, $\mathbf{W}_y^* \leftarrow \mathbf{W}_y^{t+1}$;

According to (16), we can calculate $\nabla_{\mathbf{W}_x} f(\mathbf{W}_x, \mathbf{W}_y)$ and $\nabla_{\mathbf{W}_y} f(\mathbf{W}_x, \mathbf{W}_y)$ as follows

$$\begin{aligned}\nabla_{\mathbf{W}_x} f(\mathbf{W}_x, \mathbf{W}_y) &= \mathbf{X}^T\mathbf{X}\mathbf{W}_x + \mathbf{X}^T\mathbf{Y}\mathbf{W}_y - \mathbf{X}^T\mathbf{Y} \\ &\quad + \alpha\mathbf{W}_x\mathbf{R} \\ \nabla_{\mathbf{W}_y} f(\mathbf{W}_x, \mathbf{W}_y) &= \mathbf{Y}^T\mathbf{Y}\mathbf{W}_y + \mathbf{Y}^T\mathbf{X}\mathbf{W}_x - \mathbf{Y}^T\mathbf{Y}\end{aligned} \tag{18}$$

Since there are two parameters, we can solve them alternatively. With fixed $\mathbf{W}_y$, according to (8), (16) and (17), the coefficient matrix $\mathbf{W}_x$ can be optimized by

$$\begin{aligned}\mathbf{W}_x^* &= \arg\min_{\mathbf{W}_x} Q_{L_f}(\mathbf{W}_x, \mathbf{W}_x^{(t)}) \\ &= \arg\min_{\mathbf{W}_x} \tfrac{L_f}{2}\|\mathbf{W}_x - \mathbf{G}_x^{(t)}\|_F^2 + g(\mathbf{W}_x, \mathbf{W}_y) \\ &= \arg\min_{\mathbf{W}_x} \tfrac{1}{2}\|\mathbf{W}_x - \mathbf{G}_x^{(t)}\|_F^2 + \tfrac{\beta}{L_f}\|\mathbf{W}_x\|_1\end{aligned} \tag{19}$$

where $\mathbf{G}_x^{(t)} = \mathbf{W}_x^{(t)} - \frac{1}{L_f}\nabla_{\mathbf{W}_x} f(\mathbf{W}_x, \mathbf{W}_y)$. The solution $\mathbf{W}_x^{t+1}$ can be obtained according to Proposition 3.1

$$\mathbf{W}_x^{t+1} = S_{\frac{\beta}{L_f}}[\mathbf{G}_x^{(t)}] \tag{20}$$

Similar to the optimization of $\mathbf{W}_x$, the optimal solution of $\mathbf{W}_y$ with fixed $\mathbf{W}_x$ can be inferred by

$$\begin{aligned}\mathbf{W}_y^* &= \arg\min_{\mathbf{W}_y} Q_{L_f}(\mathbf{W}_y, \mathbf{W}_y^{(t)}) \\ &= \arg\min_{\mathbf{W}_y} \tfrac{L_f}{2}\|\mathbf{W}_y - \mathbf{G}_y^{(t)}\|_F^2 + g(\mathbf{W}_x, \mathbf{W}_y) \\ &= \arg\min_{\mathbf{W}_y} \tfrac{1}{2}\|\mathbf{W}_y - \mathbf{G}_y^{(t)}\|_F^2 + \tfrac{\gamma}{L_f}\|\mathbf{W}_y\|_1\end{aligned} \tag{21}$$

where $\mathbf{G}_y^{(t)} = \mathbf{W}_y^{(t)} - \frac{1}{L_f}\nabla_{\mathbf{W}_y} f(\mathbf{W}_x, \mathbf{W}_y)$. The solution $\mathbf{W}_y^{t+1}$ can be obtained according to Proposition 3.1

$$\mathbf{W}_y^{t+1} = S_{\frac{\gamma}{L_f}}[\mathbf{G}_y^{(t)}] \tag{22}$$

We summarize all the procedures of optimization for LLSF-DL via accelerated proximal gradient method in Algorithm 2.

Algorithm 3: Testing of LLSF

Input: Test data matrix $\mathbf{X}_t \in \mathbb{R}^{m\times p}$; Coefficient matrix $\mathbf{W} \in \mathbb{R}^{p\times l}$; Threshold τ.
Output: Predicted Label Matrix: $\mathbf{Y}_t$; Score Matrix: $\mathbf{S}_t$.

$\mathbf{S}_t \leftarrow \mathbf{X}_t\mathbf{W}$;
$\mathbf{Y}_t \leftarrow \text{sign}(\mathbf{S}_t - \tau)$;

3.8 Lipschitz Constant

In this section, we will present how to calculate the *Lipschitz* constant for LLSF and LLSF-DL.

For LLSF, given $\mathbf{W}_1$ and $\mathbf{W}_2$, according to (13), then we have

$$\begin{aligned}&\|\nabla f(\mathbf{W}_1) - \nabla f(\mathbf{W}_2)\|_F^2 \\ &\quad \le (2\|\mathbf{X}^T\mathbf{X}\|_2^2 + 2\|\alpha\mathbf{R}\|_2^2)\|\Delta\mathbf{W}\|_F^2\end{aligned}$$

where $\Delta\mathbf{W} = \mathbf{W}_1 - \mathbf{W}_2$. Therefore, the *Lipschitz* constant is

$$L_f = \sqrt{2\|\mathbf{X}^T\mathbf{X}\|_2^2 + 2\|\alpha\mathbf{R}\|_2^2} \tag{23}$$

For LLSF-DL, given $\mathbf{W}^1 = (\mathbf{W}_x^1, \mathbf{W}_y^1)$ and $\mathbf{W}^2 = (\mathbf{W}_x^2, \mathbf{W}_y^2)$, according to (18), we have

$$\begin{aligned}&\|\nabla f(\mathbf{W}^1) - \nabla f(\mathbf{W}^2)\|_F^2 \\ &\le (3(\|\mathbf{X}^T\mathbf{X}\|_2^2 + \|\mathbf{X}^T\mathbf{Y}\|_2^2 + \|\alpha\mathbf{R}\|_2^2) + 2\|\mathbf{Y}^T\mathbf{Y}\|_2^2)\left\| \begin{matrix}\Delta\mathbf{W}_x \\ \Delta\mathbf{W}_y\end{matrix} \right\|_F^2\end{aligned}$$

where $\Delta\mathbf{W}_x = \mathbf{W}_x^1 - \mathbf{W}_x^2$ and $\Delta\mathbf{W}_y = \mathbf{W}_y^1 - \mathbf{W}_y^2$. Therefore, the *Lipschitz* constant is

$$L_f = \sqrt{3(\|\mathbf{X}^T\mathbf{X}\|_2^2 + \|\mathbf{X}^T\mathbf{Y}\|_2^2 + \|\alpha\mathbf{R}\|_2^2) + 2\|\mathbf{Y}^T\mathbf{Y}\|_2^2} \tag{24}$$

The proofs of *Lipschitz* continuity of (11) and (16) are provided in the supplementary material.

3.9 Testing of LLSF

After training of LLSF, we can obtain the coefficient matrix $\mathbf{W}$. Given a testing data $\mathbf{X}_t$, the predicted labels can be determined by $\text{sign}(\mathbf{S}_t - \tau)$ with the given threshold τ, where $\mathbf{S}_t = \mathbf{X}_t\mathbf{W}$. In our experiments, τ is set to be 0.5. The details of testing of LLSF are summarized in Algorithm 3.

LLSF can be utilized as a feature selection method for multi-label classification, where features corresponding to the non-zero components of $\mathbf{w}^i$ are considered as the *label-specific features* of label y_i. It can be taken as the input of many existing multi-label classification algorithms comprising a number of binary classifiers, such as BR [3], LIFT [19], and ECC [24]. The procedures are summarized in Algorithm 4.

3.10 Testing of LLSF-DL

Similar to CC and *stacking* approaches, the ground truth label $\mathbf{Y}$ is available as additional features to induce each binary classifier of LLSF-DL during the training stage[1], whereas this information is not available for new instances to be classified. Consequently, in order to make LLSF-DL applicable, the help of some other (multi-label) classifiers is needed to estimate $\hat{\mathbf{Y}}_t$ first, which can then be used in place of $\mathbf{Y}_t$ in the test stage.

In this paper, $\hat{\mathbf{Y}}_t \in \{0,1\}^{m\times l}$ is first predicted by LLSF, and then augmented with the data matrix $\mathbf{X}_t$ for prediction. The prediction of LLSF-DL can be obtained by $\hat{\mathbf{Y}}_t = \text{sign}(\mathbf{S}_t - \tau)$ with

1. A very recent work [47] has identified this discrepancy between training and testing as problematic in stacking and chaining approaches and suggests the use of out-of-sample estimates (instead of ground truth labels) at training time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering

Algorithm 4: LLSF as a Feature Selection Method

Input: Training data matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$,label matrix $\mathbf{Y} \in \mathbb{R}^{n\times l}$, and weighting parameters α, β, ρ; Test data matrix $\mathbf{X}_t \in \mathbb{R}^{m\times p}$; Binary classifier learning function $f(\cdot)$ and its parameters $\boldsymbol{\theta}$;
Output: Predicted Label Matrix: $\mathbf{Y}_t$; Score Matrix: $\mathbf{S}_t$.

Training:
learn the coefficient matrix $\mathbf{W}$ of LLSF by Algorithm 1;
for i = 1 *to* l **do**
  $\mathbf{d}^i \leftarrow \text{find}(\mathbf{w}^i \neq 0)$;
  $\mathbf{X}_i \leftarrow \mathbf{X}(:, \mathbf{d}^i)$;
  $h_i \leftarrow f(\mathbf{X}_i, \mathbf{Y}, \boldsymbol{\theta})$;
Test:
for i = 1 *to* l **do**
  $\mathbf{X}_t^i \leftarrow \mathbf{X}_t(:, \mathbf{d}^i)$;
  $[\mathbf{y}_t^i, \mathbf{s}_t^i] \leftarrow h_i(\mathbf{X}_t^i, \mathbf{Y}_t, \boldsymbol{\theta})$;
  $\mathbf{Y}_t(:, i) \leftarrow \mathbf{y}_t^i$;
  $\mathbf{S}_t(:, i) \leftarrow \mathbf{s}_t^i$;

Algorithm 5: Testing of LLSF-DL

Input: Test data matrix $\mathbf{X}_t \in \mathbb{R}^{m\times p}$; Coefficient matrix $\mathbf{W}_x \in \mathbb{R}^{p\times l}$ and $\mathbf{W}_y \in \mathbb{R}^{l\times l}$; Threshold τ. Parameters $\boldsymbol{\theta}$ and k
Output: Predicted Label Matrix: $\mathbf{Y}_t$; Score Matrix: $\mathbf{S}_t$.

$\hat{\mathbf{Y}}_t \leftarrow \text{LLSF}(\mathbf{X}_t, \boldsymbol{\theta}, \tau)$;
for i = 1 *to* k **do**
  $\mathbf{S}_t \leftarrow \mathbf{X}_t\mathbf{W}_x + \hat{\mathbf{Y}}_t\mathbf{W}_y$;
  $\hat{\mathbf{Y}}_t \leftarrow \text{sign}(\mathbf{S}_t - \tau)$;
$\mathbf{Y}_t \leftarrow \hat{\mathbf{Y}}_t$;

the given threshold τ, where $\mathbf{S}_t = \mathbf{X}_t\mathbf{W}_x + \hat{\mathbf{Y}}_t\mathbf{W}_y$. Intuitively, the more accurate the prediction of $\hat{\mathbf{Y}}_t$ by LLSF is, the more accurate of $\hat{\mathbf{Y}}_t$ predicted by LLSF-DL is. Moreover, LLSF-DL is expected to achieve better performance than LLSF. Thus, once $\hat{\mathbf{Y}}_t$ is obtained, it can be used to fine-tune the prediction by itself. In the experiment, we found that the result of $\hat{\mathbf{Y}}_t$ can be converged within 3 times. The details are summarized in Algorithm 5.

3.11 Time Complexity

The complexity of LLSF has two parts: initializations and iterations. To initialize $\mathbf{W}_1$, the calculation consists of some operations of matrix multiplication and inversion. This leads to a complexity of $\mathcal{O}(d^2n+d^3+dnl+d^2l)$, where d is the dimensionality of a data set, l is the number of labels and n is the number of instances. The initialization of L_f refers to singular value decomposition, whose complexity is $\mathcal{O}(d^3+l^3)$. In the iterations, the time cost is dominated by step 6. It needs to calculate the gradient of $f(\mathbf{W})$, which leads to a complexity of $\mathcal{O}(d^2l+dl^2)$. The total complexity for the iterations is $\mathcal{O}(t(d^2l+dl^2))$, where t is the number of iterations. Similarly, the complexity of LLSF-DL also has two parts (see Algorithm 2). In the initialization stage, the time complexity is $\mathcal{O}(d^2n+d^3+dnl+d^2l)$ + $\mathcal{O}(l^2n+l^3)$ + $\mathcal{O}(d^3+l^3+d^2l)$. The iterations need $\mathcal{O}(t(d^2l+dl^2+l^3))$. Both LLSF and LLSF-DL need memory of $\mathcal{O}(d^2+dl+l^2+nd+nl)$.

4 Experiments

4.1 Data Sets

We conduct experiments on fifteen multi-label benchmark data sets [48], [49], [50], [51], [52], [53], [54], and the details of which are summarized in Table 1. There are twelve regular-scale and three relative large-scale data sets (i.e. *bookmark*, *imdb*, and *nuswide*). They can be downloaded from the websites of Mulan[2] [10], Lamda[3] [13] and Meka[4] [55]. Column "Cardinality" means the average number of labels per example of a data set.

TABLE 1
Description of Data Sets

Data set	# Instances	# Features	# Labels	Cardinality	Domain
CAL500	502	68	174	26.044	music
genbase	645	1186	27	1.252	biology
medical	978	1449	45	1.245	text
language log	1459	1004	75	1.180	text
corel5k	5000	499	374	3.522	image
arts	5000	462	26	1.636	text(web)
education	5000	550	33	1.461	text(web)
recreation	5000	606	22	1.423	text(web)
science	5000	743	40	1.451	text(web)
rcv1(subset1)	6000	944	101	2.880	text
bibtex	7395	1836	159	2.402	text
corel16k001	13766	500	153	2.859	image
bookmark	87856	2150	208	2.028	text
imdb	120919	1001	28	2.000	text
nuswide	269468	500	81	1.869	image

4.2 Comparison Methods

We compare our proposed methods LLSF and LLSF-DL with the following state-of-the-art multi-label classification methods.

BR [3]: Binary Relevance decomposes a multi-label classification problem into l independent binary classification subproblems.

DBR [27]: Dependent Binary Relevance models high-order label correlations in a stacking way.

ECC [24]: Ensemble Classifier Chains. It is an ensemble version of CC, where the ensemble size m is set to be 10. The chain order $y_{\pi(1)}, y_{\pi(2)}, ..., y_{\pi(l)}$ for each CC is generated randomly.

MCC [29]: Efficient monte carlo methods for multi-dimensional learning with classifier chains. It is implemented in Meka [55]. SVM fitted with logistic models in the SMO implementation with polynomial kernel is utilized as the base binary learner for each binary classifier of MCC, and C is tuned in $\{10^{-4}, 10^{-3}, ..., 10^4\}$.

LIFT[5] [19]: It exploits different features set for the discrimination of different class labels. The ratio parameter r is tuned in $\{0.1, 0.2, ..., 0.5\}$.

Lasso [45]: It learns sparse label-specific features without considering label correlations. The regularizer parameter is searched in $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$.

GFLasso[6] [22]: Graph-guide Fused Lasso. The two regularization parameters are tuned in $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$.

LLSF[7]: Parameters α, β are searched in $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$, and ρ is searched in $\{0.1, 1, 10\}$. The threshold $\tau = 0.5$. The *Lipschitz* constant L_f is calculated by (23).

LLSF-DL: Parameters α, β, γ are searched in $\{4^{-5}, 4^{-4}, ..., 4^5\}$, and ρ is searched in $\{0.1, 1, 10\}$. The threshold $\tau = 0.5$ and $k = 3$. The *Lipschitz* constant L_f is calculated by (24).

LLSF-BR: The data composed of *label-specific features* learned by LLSF of each label is set to be the training data for each corresponding binary classifier of BR.

LLSF-ECC: The ensemble size m is set to be 10, and the chains order $y_{\pi(1)}, y_{\pi(2)}, ..., y_{\pi(l)}$ is generated randomly. Each classifier $h_{\pi(i)}$ is trained by using $y_{\pi(1)}, y_{\pi(2)}, ..., y_{\pi(i-1)}$ as augmented features with the *label-specific features* of label $y_{\pi(i)}$.

2. http://mulan.sourceforge.net/datasets.html
3. http://lamda.nju.edu.cn/Data.ashx#data
4. http://meka.sourceforge.net
5. source code: http://cse.seu.edu.cn/PersonalPage/zhangml/index.htm
6. source code: http://www.cs.cmu.edu/%7Esssykim/softwares/softwares.html
7. source code: http://www.escience.cn/people/huangjun/index.html.

TABLE 2
Experimental results of each comparing algorithm (mean±std) in terms of *Hamming Loss, Accuracy, and Exact Match*. ↑ (↓) indicates the larger (smaller) the value, the better the performance. Best results are highlighted in **bold**

Data Set	*Hamming Loss*↓								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	**0.137**±0.004	0.257±0.010	**0.137**±0.003	0.138±0.003	0.168±0.008	0.204±0.006	0.138±0.005	0.144±0.005	0.238±0.041
genbase	**0.001**±0.001	**0.001**±0.000	**0.001**±0.001	**0.001**±0.001	**0.001**±0.001	**0.001**±0.001	0.002±0.001	**0.001**±0.000	**0.001**±0.000
medical	0.011±0.001	**0.010**±0.001	0.011±0.001	**0.010**±0.001	0.011±0.001	**0.010**±0.001	0.011±0.001	**0.010**±0.001	**0.010**±0.002
language log	0.021±0.004	0.016±0.001	0.016±0.001	0.045±0.002	0.016±0.001	**0.015**±0.001	0.016±0.001	0.018±0.001	0.019±0.000
corel5k	**0.009**±0.000	0.010±0.000	**0.009**±0.000	**0.009**±0.000	0.019±0.000	0.017±0.000	0.012±0.000	0.012±0.000	0.011±0.000
arts	0.054±0.001	0.053±0.001	0.053±0.002	0.166±0.019	0.056±0.001	0.060±0.001	**0.052**±0.001	0.057±0.002	0.058±0.002
education	0.038±0.001	0.053±0.001	0.037±0.001	0.116±0.007	0.041±0.002	0.050±0.003	**0.036**±0.000	0.042±0.001	0.043±0.001
recreation	0.054±0.001	0.054±0.002	**0.052**±0.001	0.195±0.012	0.055±0.005	0.061±0.002	**0.052**±0.001	0.055±0.001	0.061±0.002
science	0.031±0.001	0.034±0.000	**0.030**±0.001	0.153±0.005	0.032±0.001	0.035±0.000	**0.030**±0.001	0.035±0.001	0.038±0.001
rcv1(subset1)	**0.026**±0.001	**0.026**±0.000	**0.026**±0.000	0.037±0.001	0.029±0.000	0.034±0.000	**0.026**±0.000	0.029±0.000	0.029±0.001
bibtex	**0.012**±0.000	**0.012**±0.000	**0.012**±0.000	0.015±0.000	0.013±0.000	0.015±0.000	**0.012**±0.000	0.013±0.000	0.014±0.000
corel16k001	**0.019**±0.000	**0.019**±0.000	**0.019**±0.000	0.031±0.000	0.021±0.001	0.020±0.000	**0.019**±0.000	0.023±0.000	0.023±0.000
Data Set	*Accuracy* ↑								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	0.201±0.004	0.140±0.010	0.190±0.008	0.197±0.004	0.211±0.010	0.201±0.005	0.194±0.011	0.263±0.014	**0.307**±0.015
genbase	0.986±0.008	0.992±0.006	0.987±0.010	0.984±0.007	0.990±0.007	0.984±0.012	0.974±0.008	0.993±0.004	**0.994**±0.005
medical	0.726±0.027	0.738±0.025	0.742±0.025	**0.766**±0.033	0.756±0.025	0.764±0.020	0.686±0.030	0.756±0.019	0.755±0.032
language log	0.117±0.026	0.124±0.004	0.105±0.013	0.157±0.008	0.128±0.012	**0.219**±0.028	0.126±0.014	0.145±0.012	0.172±0.013
corel5k	0.055±0.006	0.056±0.004	0.039±0.006	0.049±0.003	0.086±0.003	0.104±0.003	0.118±0.009	0.144±0.007	**0.146**±0.005
arts	0.277±0.008	0.298±0.007	0.282±0.017	0.338±0.014	0.325±0.022	**0.413**±0.028	0.301±0.013	0.374±0.017	0.375±0.012
education	0.297±0.008	0.298±0.007	0.265±0.011	0.347±0.010	0.361±0.013	**0.393**±0.031	0.312±0.009	0.368±0.014	0.376±0.020
recreation	0.301±0.018	0.297±0.011	0.295±0.005	0.361±0.006	0.376±0.028	**0.409**±0.011	0.320±0.010	0.352±0.007	0.398±0.005
science	0.268±0.011	0.260±0.005	0.278±0.012	0.315±0.006	0.362±0.010	**0.416**±0.000	0.303±0.015	0.350±0.012	0.367±0.013
rcv1(subset1)	0.252±0.011	0.255±0.010	0.253±0.006	0.271±0.009	0.339±0.008	0.320±0.006	0.277±0.008	0.351±0.005	**0.355**±0.007
bibtex	0.303±0.007	0.308±0.012	0.346±0.012	0.292±0.010	0.332±0.006	0.342±0.037	0.317±0.009	0.360±0.011	**0.383**±0.010
corel16k001	0.038±0.002	0.050±0.002	0.018±0.001	**0.161**±0.002	0.091±0.016	0.106±0.001	0.026±0.002	0.144±0.005	0.155±0.005
Data set	*Exact Match* ↑								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
genbase	0.974±0.014	**0.986**±0.009	0.971±0.022	0.967±0.015	0.982±0.015	0.968±0.016	0.958±0.014	**0.986**±0.009	0.985±0.013
medical	0.631±0.029	0.653±0.025	0.643±0.035	0.681±0.034	0.689±0.037	**0.694**±0.016	0.612±0.037	0.668±0.025	0.670±0.030
language log	0.210±0.012	0.237±0.025	0.215±0.002	0.217±0.022	0.242±0.024	0.206±0.028	0.239±0.024	0.243±0.016	**0.244**±0.022
corel5k	0.005±0.003	0.005±0.001	0.006±0.001	0.006±0.003	0.003±0.003	0.007±0.004	**0.015**±0.003	0.008±0.001	0.010±0.003
arts	0.222±0.005	0.236±0.003	0.227±0.019	0.219±0.009	0.272±0.019	**0.351**±0.025	0.241±0.012	0.267±0.019	0.269±0.012
education	0.240±0.007	0.236±0.003	0.221±0.009	0.220±0.012	**0.296**±0.015	0.294±0.035	0.253±0.005	0.254±0.011	0.256±0.018
recreation	0.259±0.016	0.256±0.012	0.257±0.006	0.257±0.008	0.335±0.027	**0.361**±0.009	0.278±0.012	0.287±0.009	0.294±0.007
science	0.220±0.012	0.199±0.006	0.233±0.012	0.229±0.007	0.317±0.011	**0.359**±0.000	0.250±0.014	0.255±0.013	0.256±0.013
rcv1(subset1)	0.079±0.007	0.081±0.006	0.083±0.004	0.098±0.007	**0.222**±0.007	0.137±0.007	0.090±0.007	0.051±0.006	0.048±0.004
bibtex	0.171±0.012	0.176±0.014	0.187±0.013	0.176±0.008	**0.189**±0.000	0.173±0.000	0.168±0.010	0.178±0.015	0.168±0.007
corel16k001	0.007±0.001	0.009±0.001	0.004±0.001	0.007±0.001	**0.019**±0.004	0.017±0.002	0.006±0.001	0.014±0.002	0.015±0.003

LLSF-LIFT: The data composed of the *label-specific features* of each class label is set to be the training data for each corresponding binary classifier of LIFT, and the ratio parameter r is tuned in $\{0.1, 0.2, ..., 0.5\}$.

For fair comparisons, libsvm [56] is utilized as the base binary learner for each binary classifier of BR, DBR, ECC, LIFT, LLSF-BR, LLSF-ECC and LLSF-LIFT, where the kernel function is set as linear kernel, and the parameter C is tuned in $\{10^{-4}, 10^{-3}, ..., 10^{4}\}$.

4.3 Results of Multi-Label Classification

We repeatedly run each comparing algorithm 5 times on 5 sets of randomly partitioned training (80%) and testing (20%) data, and the parameters are tuned by 5-fold internal cross validation on the training data. Tables 2 and 3 report the average results of each comparing algorithm over twelve regular-scale data sets in terms of six evaluation metrics (*the definition of each metric is provided in the supplementary file*).

Friedman test [57] is employed to conduct performance analysis among the comparing algorithms. Table 4 provides the Friedman statistics F_F and the corresponding critical value in terms of each evaluation metric. As shown in Table 4, at significance level $\alpha = 0.05$, the null hypothesis that all the comparing algorithms perform equivalently is clearly rejected in terms of each evaluation metric. Consequently, we can proceed with a post-hoc test [57] to analyse the relative performance among the comparing algorithms. The Nemenyi test [57] is employed to test whether our proposed method LLSF or LLSF-DL achieves a competitive performance

TABLE 4
Summary of the Friedman statistics $F_F(k = 9, N = 12)$ and the critical value in terms of each evaluation metric (k: # Comparing algorithms; N: # Data sets)

Metric	F_F	Critical Value ($\alpha = 0.05$)
Hamming Loss	4.4098	2.0454
Accuracy	18.2431	
Exact Match	4.5091	
F_1	15.2687	
Macro F_1	8.6465	
Micro F_1	19.0190	

against the comparing algorithms, where LLSF or LLSF-DL is considered as the control algorithm, respectively. The performance between two classifiers will be significantly different if the corresponding average ranks differ by at least the critical difference $\text{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$. For Nemenyi test, $q_\alpha = 3.102$ at significance level $\alpha = 0.05$, and thus $\text{CD} = 3.4681$ $(k = 9, N = 12)$. Fig. 3 shows the CD diagrams on each evaluation metric. In each subfigure, any comparing algorithm whose average rank is within one CD to that of LLSF or LLSF-DL is connected. Otherwise, any algorithm not connected with LLSF or LLSF-DL is considered to have significant different performance between them.

Based on these experimental results, the following observations can be made:

- These first order algorithms (e.g., BR, LIFT, Lasso,) obtain better performance on *hamming loss* (see Fig. 3(a))

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
Experimental results of each comparing algorithm (mean±std) in terms of F_1, *Macro* F_1, *and Micro* F_1. ↑ indicates the larger the value, the better the performance. Best results are highlighted in **bold**

Data set	F_1 ↑								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	0.330±0.005	0.240±0.017	0.314±0.010	0.324±0.006	0.341±0.013	0.325±0.009	0.319±0.016	0.410±0.017	**0.464**±0.017
genbase	0.990±0.007	0.994±0.005	0.991±0.008	0.989±0.005	0.992±0.006	0.989±0.010	0.979±0.007	0.995±0.003	**0.996**±0.004
medical	0.758±0.027	0.767±0.027	0.772±0.024	**0.796**±0.033	0.779±0.021	0.789±0.022	0.711±0.029	0.786±0.020	0.783±0.032
language log	0.136±0.034	0.134±0.004	0.115±0.016	**0.202**±0.009	0.138±0.013	0.074±0.020	0.136±0.014	0.161±0.012	0.198±0.014
corel5k	0.078±0.007	0.081±0.005	0.053±0.008	0.068±0.003	0.133±0.003	0.152±0.003	0.167±0.013	0.208±0.011	**0.209**±0.008
arts	0.297±0.009	0.321±0.008	0.303±0.016	0.408±0.014	0.345±0.023	**0.437**±0.029	0.323±0.014	0.414±0.017	0.414±0.012
education	0.317±0.009	0.321±0.008	0.280±0.012	0.419±0.010	0.373±0.014	**0.430**±0.030	0.332±0.010	0.408±0.015	0.419±0.021
recreation	0.317±0.019	0.312±0.011	0.309±0.006	0.424±0.006	0.390±0.029	0.427±0.012	0.336±0.009	0.375±0.008	**0.437**±0.006
science	0.285±0.011	0.283±0.005	0.295±0.012	0.369±0.007	0.378±0.010	**0.438**±0.000	0.322±0.015	0.384±0.013	0.412±0.014
rcv1(subset1)	0.320±0.012	0.323±0.013	0.320±0.007	0.347±0.011	0.390±0.010	0.395±0.008	0.350±0.011	0.456±0.006	**0.461**±0.008
bibtex	0.355±0.007	0.361±0.011	0.404±0.011	0.338±0.010	0.387±0.008	0.410±0.000	0.375±0.009	0.426±0.011	**0.462**±0.011
corel16k001	0.053±0.002	0.069±0.003	0.024±0.001	**0.242**±0.003	0.124±0.022	0.148±0.002	0.035±0.003	0.204±0.007	0.219±0.007
Data set	*Macro* F_1 ↑								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	0.056±0.003	0.069±0.007	0.040±0.002	0.052±0.001	0.125±0.006	**0.163**±0.001	0.039±0.002	0.068±0.007	0.153±0.020
genbase	0.721±0.019	0.738±0.064	0.730±0.030	0.676±0.045	0.750±0.033	0.705±0.068	0.705±0.017	**0.769**±0.057	0.747±0.076
medical	0.322±0.018	**0.370**±0.020	0.350±0.018	0.368±0.025	0.366±0.015	0.321±0.022	0.270±0.020	0.352±0.034	0.363±0.017
language log	0.051±0.017	0.043±0.005	0.050±0.002	0.054±0.003	0.050±0.005	0.042±0.018	0.053±0.001	0.069±0.014	**0.079**±0.006
corel5k	0.017±0.001	0.014±0.001	0.019±0.003	0.028±0.002	**0.050**±0.003	0.044±0.002	0.025±0.003	0.039±0.002	0.039±0.002
arts	0.170±0.014	0.193±0.010	0.186±0.006	0.151±0.003	0.163±0.008	0.208±0.015	0.206±0.014	0.238±0.010	**0.242**±0.012
education	0.122±0.006	0.193±0.010	0.157±0.026	0.146±0.026	0.160±0.015	**0.218**±0.021	0.175±0.009	0.186±0.011	0.183±0.021
recreation	0.228±0.004	0.228±0.013	0.248±0.016	0.192±0.012	0.262±0.012	0.269±0.016	0.274±0.009	0.274±0.011	**0.324**±0.004
science	0.146±0.012	0.142±0.011	0.169±0.007	0.147±0.004	0.183±0.006	0.204±0.000	0.188±0.014	0.213±0.016	**0.221**±0.013
rcv1(subset1)	0.131±0.006	0.124±0.005	0.189±0.009	0.194±0.011	0.202±0.011	0.247±0.010	0.209±0.010	0.251±0.004	**0.255**±0.010
bibtex	0.222±0.009	0.228±0.009	0.303±0.009	0.238±0.010	0.296±0.005	0.331±0.000	0.290±0.017	0.328±0.003	**0.368**±0.009
corel16k001	0.016±0.002	0.014±0.002	0.021±0.004	0.046±0.002	0.041±0.004	0.031±0.001	0.024±0.003	0.064±0.004	**0.069**±0.002
Data set	*Micro* F_1 ↑								
	Lasso	GFLasso	BR	DBR	ECC	MCC	LIFT	LLSF	LSFS-DL
CAL500	0.328±0.006	0.242±0.018	0.309±0.010	0.320±0.005	0.341±0.015	0.330±0.010	0.313±0.016	0.409±0.018	**0.468**±0.017
genbase	0.989±0.006	0.993±0.005	0.987±0.011	0.986±0.007	0.992±0.005	0.985±0.009	0.979±0.006	**0.994**±0.004	**0.994**±0.005
medical	0.798±0.020	0.807±0.022	0.802±0.016	0.813±0.024	0.798±0.018	0.807±0.019	0.771±0.021	**0.817**±0.016	0.815±0.032
language log	0.193±0.029	0.214±0.011	0.196±0.027	0.196±0.005	0.217±0.030	0.133±0.032	0.224±0.020	0.233±0.014	**0.278**±0.011
corel5k	0.106±0.008	0.105±0.006	0.076±0.012	0.096±0.006	0.139±0.004	0.155±0.003	0.180±0.013	0.244±0.012	**0.247**±0.006
arts	0.356±0.008	0.378±0.011	0.367±0.017	0.297±0.018	0.374±0.011	0.413±0.022	0.383±0.014	**0.445**±0.017	**0.445**±0.014
education	0.399±0.009	0.378±0.011	0.374±0.016	0.324±0.012	0.430±0.008	0.420±0.027	0.422±0.013	0.459±0.013	**0.462**±0.013
recreation	0.386±0.018	0.381±0.015	0.384±0.007	0.291±0.006	0.425±0.013	0.423±0.005	0.405±0.009	0.430±0.010	**0.462**±0.003
science	0.361±0.011	0.341±0.006	0.374±0.012	0.225±0.005	0.416±0.007	0.431±0.000	0.400±0.016	0.436±0.014	**0.441**±0.014
rcv1(subset1)	0.365±0.014	0.366±0.011	0.366±0.006	0.335±0.011	0.395±0.008	0.402±0.006	0.387±0.012	0.495±0.005	**0.500**±0.006
bibtex	0.406±0.007	0.411±0.008	0.461±0.014	0.385±0.010	0.431±0.002	0.430±0.000	0.434±0.011	0.490±0.008	**0.495**±0.009
corel16k001	0.071±0.003	0.086±0.004	0.036±0.004	0.249±0.003	0.143±0.021	0.168±0.001	0.048±0.004	0.243±0.007	**0.258**±0.006

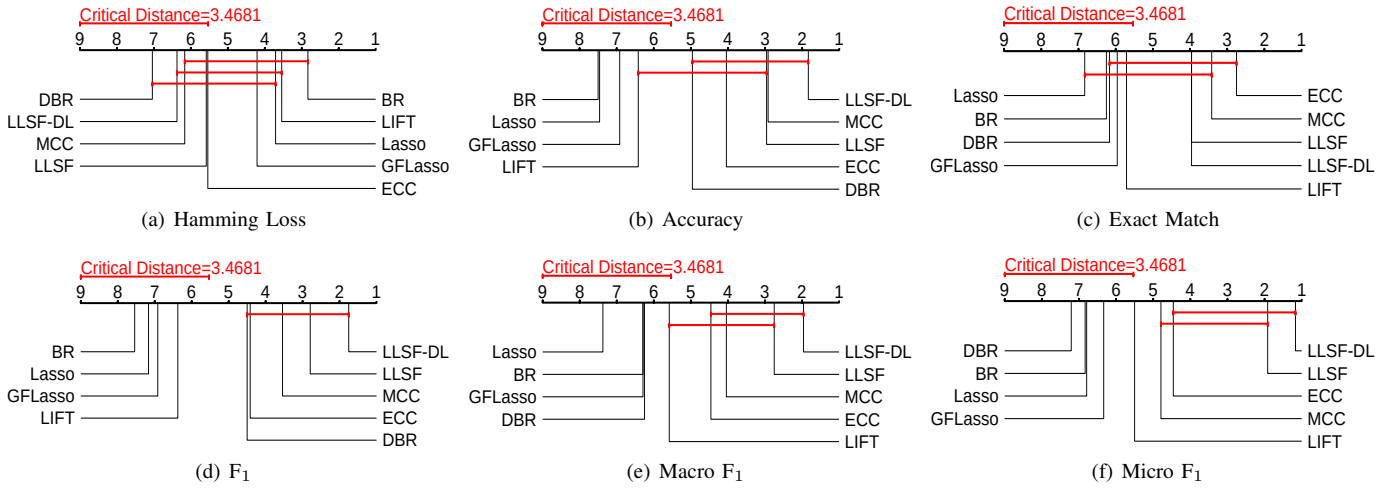


Fig. 3. Comparison of LLSF or LLSF-DL (control algorithm) against other comparing algorithms with the Nemenyi test. Groups of classifiers that are not significantly different from LLSF or LLSF-DL (at $p = 0.05$) are connected.

than these algorithms which incorporate *second-order* (e.g., LLSF, GFLasso) or *high-order* (e.g., ECC, MCC, LLSF-DL, DBR) label correlations, as *first-order* algorithms try to optimize hamming loss.

- While on *exact match* (see Fig. 3(c)), the algorithms, which incorporate *second-order* (e.g., GFLasso, LLSF) or *high-order* (e.g., ECC, MCC, LLSF-DL, DBR) label correlations, obtain better performance than the first order algorithms (e.g., BR and Lasso). As previous researches suggest that optimizing exact match need to model label correlations.
- LLSF performs better than DBR and MCC, and worse than other comparing algorithms in terms of *hamming loss*. On *exact match*, LLSF performs worse than ECC and MCC, but better than other comparing algorithms. LLSF significantly

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering

TABLE 5
Experimental results of BR, LIFT and ECC after using label-specific features learned by LLSF (mean±std) in terms of each evaluation metric. ↑ (↓) indicates the larger (smaller) the value, the better the performance

Data Set	*Hamming Loss*↓			*Accuracy*↑			*Exact Match*↑		
	LLSF-BR	LLSF-LIFT	LLSF-ECC	LLSF-BR	LLSF-LIFT	LLSF-ECC	LLSF-BR	LLSF-LIFT	LLSF-ECC
CAL500	0.138±0.004	0.146±0.009	0.138±0.002	0.192±0.006	0.191±0.012	0.200±0.013	0.000±0.000	0.000±0.000	0.000±0.000
genbase	0.001±0.000	0.001±0.000	0.001±0.000	0.988±0.004	0.985±0.005	0.988±0.005	0.976±0.011	0.967±0.010	0.976±0.011
medical	0.010±0.001	0.011±0.001	0.010±0.001	0.755±0.022	0.718±0.026	0.766±0.014	0.669±0.034	0.640±0.031	0.691±0.024
language log	0.017±0.001	0.016±0.001	0.018±0.001	0.154±0.020	0.138±0.018	0.169±0.015	0.245±0.026	0.240±0.030	0.278±0.026
corek5k	0.009±0.000	0.009±0.000	0.011±0.001	0.053±0.006	0.047±0.006	0.101±0.016	0.009±0.004	0.007±0.003	0.017±0.003
arts	0.052±0.002	0.052±0.001	0.062±0.002	0.294±0.019	0.294±0.002	0.389±0.026	0.237±0.016	0.238±0.004	0.323±0.028
education	0.037±0.000	0.035±0.001	0.043±0.003	0.284±0.010	0.300±0.007	0.381±0.019	0.235±0.008	0.254±0.005	0.317±0.017
recreation	0.053±0.001	0.052±0.001	0.066±0.002	0.317±0.005	0.311±0.010	0.409±0.024	0.274±0.005	0.266±0.007	0.361±0.023
science	0.031±0.001	0.032±0.000	0.038±0.001	0.304±0.019	0.281±0.009	0.383±0.005	0.249±0.015	0.223±0.008	0.327±0.010
rcv1(subset1)	0.026±0.001	0.025±0.001	0.028±0.000	0.293±0.014	0.281±0.005	0.355±0.008	0.101±0.011	0.089±0.007	0.226±0.008
bibtex	0.011±0.000	0.013±0.000	0.012±0.000	0.373±0.007	0.341±0.005	0.358±0.008	0.211±0.003	0.178±0.006	0.221±0.007
corel16k001	0.019±0.000	0.019±0.000	0.021±0.001	0.018±0.003	0.020±0.001	0.105±0.023	0.004±0.001	0.003±0.001	0.020±0.003

Data Set	F_1↑			*Macro* F_1↑			*Micro* F_1↑		
	LLSF-BR	LLSF-LIFT	LLSF-ECC	LLSF-BR	LLSF-LIFT	LLSF-ECC	LLSF-BR	LLSF-LIFT	LLSF-ECC
CAL500	0.317±0.009	0.316±0.016	0.327±0.018	0.039±0.000	0.052±0.004	0.041±0.004	0.312±0.008	0.314±0.016	0.321±0.018
genbase	0.991±0.003	0.989±0.004	0.991±0.003	0.734±0.019	0.696±0.026	0.726±0.013	0.990±0.005	0.985±0.005	0.990±0.005
medical	0.784±0.019	0.745±0.025	0.791±0.011	0.356±0.013	0.317±0.011	0.362±0.017	0.814±0.015	0.789±0.024	0.809±0.017
language log	0.173±0.019	0.153±0.020	0.182±0.016	0.088±0.010	0.075±0.016	0.076±0.007	0.263±0.025	0.240±0.029	0.245±0.022
corel5k	0.072±0.007	0.064±0.008	0.138±0.023	0.025±0.003	0.023±0.003	0.030±0.004	0.104±0.011	0.092±0.011	0.161±0.029
arts	0.316±0.019	0.315±0.001	0.415±0.026	0.206±0.015	0.192±0.006	0.205±0.009	0.379±0.020	0.379±0.004	0.399±0.008
education	0.302±0.010	0.316±0.008	0.404±0.020	0.141±0.009	0.126±0.013	0.181±0.015	0.389±0.012	0.405±0.009	0.424±0.024
recreation	0.332±0.005	0.326±0.011	0.427±0.025	0.247±0.003	0.259±0.012	0.265±0.017	0.401±0.007	0.400±0.011	0.406±0.017
science	0.323±0.020	0.301±0.009	0.404±0.004	0.178±0.011	0.179±0.004	0.193±0.019	0.397±0.021	0.373±0.010	0.394±0.006
rcv1(subset1)	0.366±0.017	0.354±0.005	0.408±0.009	0.212±0.017	0.227±0.009	0.212±0.015	0.405±0.020	0.413±0.009	0.414±0.012
bibtex	0.432±0.007	0.404±0.006	0.408±0.008	0.332±0.009	0.297±0.006	0.301±0.006	0.481±0.013	0.446±0.004	0.461±0.010
corel16k001	0.023±0.004	0.027±0.002	0.146±0.034	0.021±0.003	0.018±0.003	0.039±0.002	0.035±0.005	0.038±0.003	0.167±0.033

TABLE 6
Wilcoxon Signed-Ranks Test for BR, LIFT, ECC againt their LLSF versions in terms of each evaluation metric (significance level $\alpha = 0.05$, $p-$values shown in the brackets)

Comparing Algorithm	*Hamming Loss*	*Accuracy*	*Exact Match*	F_1	*Macro* F_1	*Micro* F_1
LLSF-BR vs BR	tie[1e-0]	win[9.8e-4]	win[2e-3]	win[2e-3]	tie[3.7e-2]	win[9.8e-4]
LLSF-LIFT vs LIFT	tie[1e-0]	tie[1e-0]	tie[9.8e-1]	tie[1e-0]	tie[8.1e-1]	tie[9.8e-1]
LLSF-ECC vs ECC	tie[1.4e-1]	win[3.9e-3]	win[9.8e-3]	win[1.6e-2]	tie[6.5e-2]	tie[2.3e-1]
LLSF-BR vs LIFT	tie[7.3e-1]	tie[3.5-1]	tie[2.9e-1]	tie[4.1e-1]	tie[6.4-1]	tie[4.4e-1]

outperforms BR, Lasso and GFLasso, achieves statistically superior or at least comparable performance against LIFT, DBR and ECC, and comparable performance against MCC in terms of the other four evaluation metrics.

- LLSF-DL performs better than DBR and worse than other comparing algorithms in terms of *hamming loss*. On *exact match*, LLSF-DL performs worse than ECC and MCC, but better than other comparing algorithms. Furthermore, LLSF-DL significantly outperforms BR, Lasso, GFLasso and LIFT, and achieves statistically superior or at least comparable performance against DBR, ECC and MCC in terms of the other four evaluation metrics.
- Furthermore, after modeling *high-order* label correlations, LLSF-DL achieves better performance than LLSF except on hamming loss.

Thus, our proposed methods achieve a competitive performance against other well-established multi-label classification algorithms.

4.4 Results of Feature Selection

LLSF can be applied as a feature selection method for multi-label learning and a general strategy to improve multi-label classification algorithms comprising a number of binary classifiers. We evaluate the performance of feature selection of LLSF in this section.

Firstly, we combine LLSF with BR [3], LIFT [19] and ECC [24], and denote them as LLSF-BR, LLSF-LIFT and LLSF-ECC, respectively. The data composed of label-specific features learned by LLSF is used as the input for each corresponding binary classifier of them. Table 5 reports the experimental results of LLSF-BR, LLSF-LIFT and LLSF-ECC. To test whether the LLSF extended version performs significantly better than its original version, the Wilcoxon signed-ranks test [57] is employed. From Tables 2, 3 and 5, we summarize the statistical test results at significance level $\alpha = 0.05$ and report them in Table 6. We can see that LLSF-BR and LLSF-ECC achieve statistically superior or at least comparable performance against their original versions. The superior performance of LLSF-BR and LLSF-ECC clearly verifies the effectiveness of employing label-specific features. LLSF-LIFT is tied with LIFT, and LLSF-BR is tied with LIFT. However, the low dimensional data representation learned by LLSF will make LIFT and BR more efficient than using all the features (*the execution time of each comparing algorithm is provided in the supplementary file*).

Secondly, we compare LLSF with Lap-Score (Laplacian Score), F-Score (Fisher Score), IG (Information Gain) and GFLasso. For Lap-Score, the similarity matrix is calculated by cosine similairity on the label matrix of the training data, and k is searched in {5, 15, 30} to construct the kNN graph. For IG, continuous data is discretized to two quantization levels by a quantile method. Parameters for GFLasso and LLSF are searched by 5-fold cross validation on the training data. Libsvm [56] with linear kernel is initialized as the binary classifier for each label, and the parameter C is tuned in $\{10^{-4}, 10^{-3}, ..., 10^{4}\}$. Experiments are conducted on genbase, medical, language log and bibtex, where the number of features is larger than 1000. Fig. 4 shows the results of the comparing algorithms in terms of six evaluation metrics when using the top 10%*p, 20%*p, ..., 50%*p features selected by each

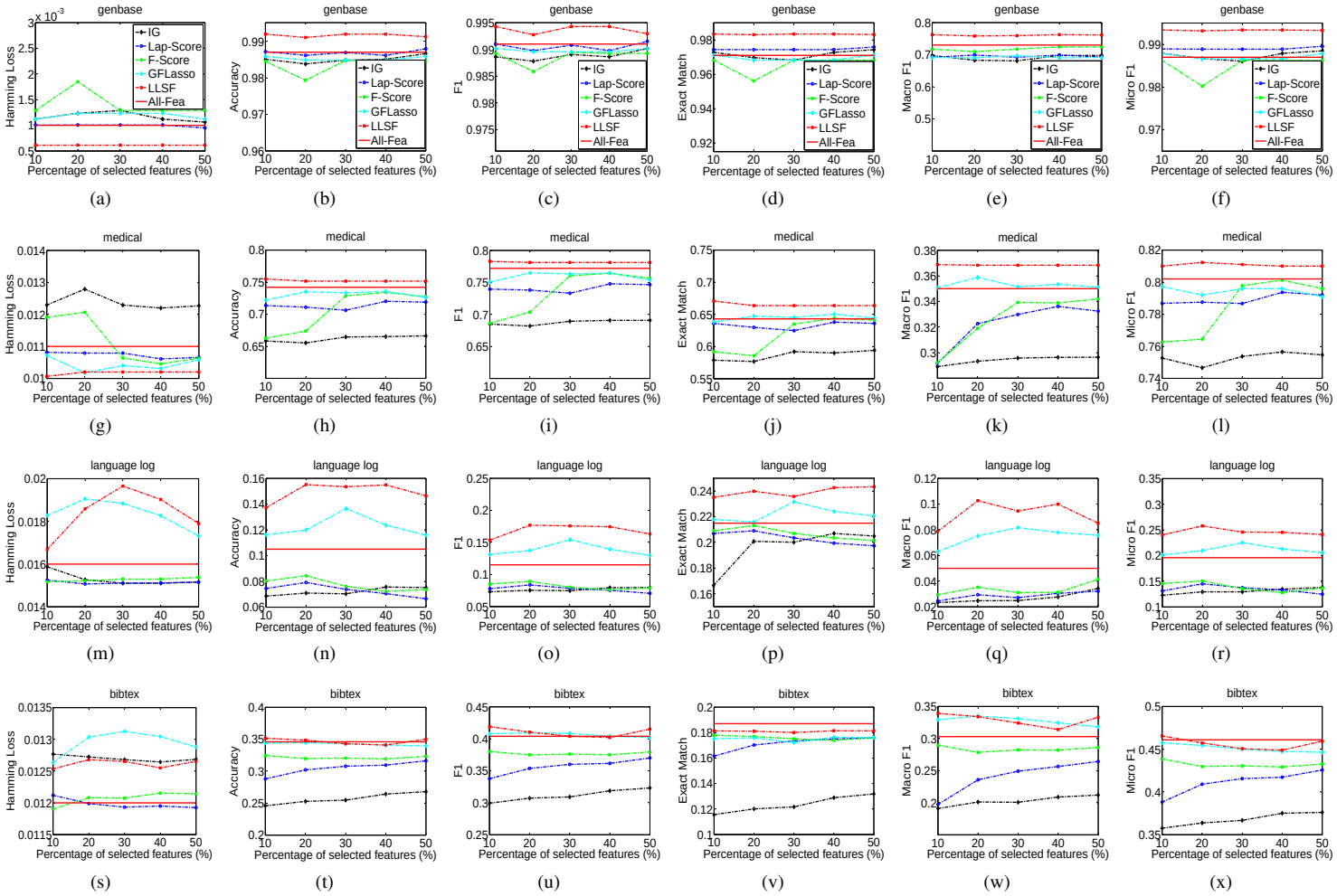


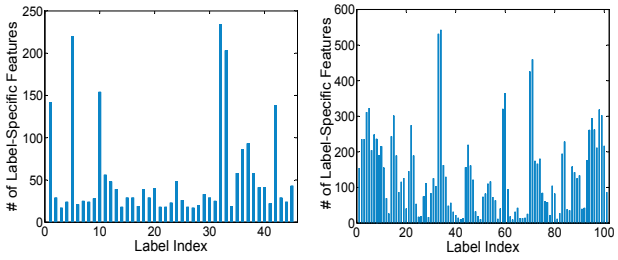
Fig. 4. Results of feature selection: BR with SVM as the base binary classifier is employed as the multi-label classifier.

algorithm for each data set, where p is the dimensionality of each data set. "All-Fea" means that the original data with no feature selection is used as a baseline, and its performance is equal to BR's. As shown in Fig. 4, LLSF achieves better performance against all the comparing algorithms in terms of each evaluation metric in most cases. These results verify the effectiveness of feature selection of LLSF.

4.5 Property of Coefficient Matrix W in LLSF

We assume that label-specific feature has three properties: *discriminability*, *sparsity* and *sharing*. The experimental results presented in Section 4.3 and Section 4.4 have shown the strong discriminability of *label-specific features*. Fig. 5 shows an example of sparsity of label-specific features on medical and rcv1(subset1) data sets. The horizontal axis of each sub-figure indicates the index of class label. The vertical axis of each sub-figure indicates the number of *label-specific features* (i.e. $\|\mathbf{w}^i\|_0$) of each class label. It can be seen that each label is only associated with a small number of relevant features from the original features set. It is controlled by the parameter β, the larger the value of β is, the more sparse of the *label-specific features* is.

On the other hand, we assume that any two strongly correlated class labels can *share* more features with each other than two uncorrelated or weakly correlated ones. In our model, the label correlation matrix $\mathbf{C}$ on $\mathbf{Y}$ is calculated by cosine similarity. Each element of $\mathbf{C}$ is defined as $c_{ij} = \frac{<\mathbf{y}^i,\mathbf{y}^j>}{\|\mathbf{y}^i\|\times\|\mathbf{y}^j\|}$, where $\mathbf{y}^i$ is the i-th column of a label matrix $\mathbf{Y}$. Exactly, the number of shared features between two labels should be calculated by $s_{ij} = \sum_{k=1}^{p} \|w_{ki}\|_0 \|w_{kj}\|_0$. The larger the value of c_{ij}, the larger the value of s_{ij}, and vice versa. For the convenient of optimization, we approximate it by calculating $s_{ij} = {\mathbf{w}^i}^T \mathbf{w}^j$.

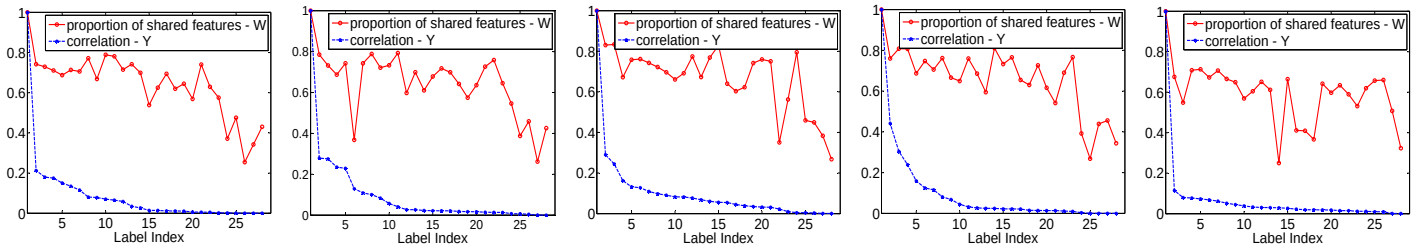


(a) result of medical (b) result of rcv1(subset1)

Fig. 5. The number of label-specific features learned by LLSF for each class label with parameters $\alpha = 2^{-1}, \beta = 2^{-1}$ and $\rho = 0.1$.

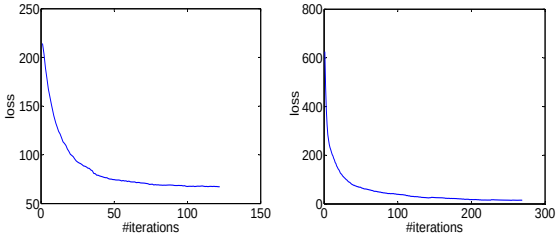
Fig. 6 shows the correlation between class labels *w.r.t* the proportion of shared features between them for five selected class labels of imdb data. The proportion of shared features between two labels is calculated by $s_{ij} = \frac{\sum_{k=1}^{p} \|w_{ki}\|_0 \|w_{kj}\|_0}{\|\mathbf{w}^i\|_0 + \|\mathbf{w}^j\|_0 - \sum_{k=1}^{p} \|w_{ki}\|_0 \|w_{kj}\|_0}$. The horizontal axis of each sub-figure indicates the index of each label, and the vertical axis indicates the value of correlation or proportion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2608339, IEEE Transactions on Knowledge and Data Engineering



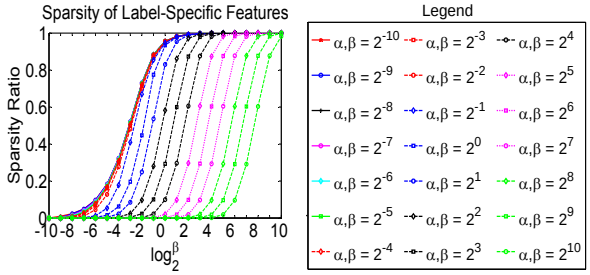
(a) y_1 to other labels (b) y_2 to other labels (c) y_3 to other labels (d) y_4 to other labels (e) y_5 to other labels

Fig. 6. Correlation *w.r.t* the proportion of shared features between class labels for five selected class labels of imdb data, and the labels are sorted in descending order by the value of label correlation.



(a) LLSF (b) LLSF-DL

Fig. 7. Loss of LLSF and LLSF-DL *w.r.t* # iterations on genbase.



(a) Sparse ratio of LLSF (b) Legend

Fig. 8. Influence of parameters α and β to the sparsity of label-specific features learned by LLSF on rcv1(subset1).

of shared features between two labels. As shown in Fig. 6, the proportion of shared features decreases with the decreasing of label correlation. This result verifies that approximately calculating s_{ij} by $\mathbf{w}^{i^T}\mathbf{w}^j$ can model the property of *sharing* well to some extent. But in some cases, the results are not consistent. One possible reason might be the semantic gap between low level features and high level class labels.

4.6 Convergence

In this paper, our methods are solved by iterative shrinkage-thresholding. In [46], the iterative shrinkage-thresholding algorithms are proven to converge in function values as $O(t^{-2})$ with an appropriate stepsize. Fig. 7 shows the value of the loss function of LLSF (4) and LLSF-DL (5) *w.r.t* the number of iterations, respectively. The value decreases dramatically, and tends to be stable after 100 iterations. The execution time of the comparing algorithms on the same computer environment is provided in the supplementary file. The experimental results show that LLSF and LLSF-DL are more efficient than most of the comparing algorithms. LLSF-BR, LLSF-ECC and LLSF-LIFT are more efficient than their standard versions after using label-specific features learned by LLSF. These results justify the efficiency of LLSF and LLSF-DL in multi-label learning.

4.7 Parameters Sensitivity Analysis

We conduct parameter sensitivity analysis for LLSF, LLSF-DL and LLSF-BR on rcv1(subset1) data over the trade-off parameters α, β, γ, and ρ, where α controls the correlation between labels, β controls the *sparsity* of *label-specific features*, γ controls the *sparsity* of *class-dependent labels*, and ρ is used for initialization of the coefficient matrix. α, β, and γ are searched in $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$, and ρ tuned in $\{10^{-3}, 10^{-2}, ..., 10^{1}\}$.

Sparsity of Label-Specific Features: Fig. 9(s) shows the influence of parameters α and β to the sparsity of *label-specific features* learned by LLSF on rcv1(subset1) data. The ratio of sparsity is calculated by $\frac{\#\text{Zero components of } \mathbf{W}}{p*l}$, where p is the dimensionality of a data set, and l is the number of class labels. Given α, the larger the β is, the larger the ratio of sparsity is. While with the growing of α, the ratio of sparsity goes down. LLSF and its extended versions will be more efficient with a larger ratio of sparsity .

Performance of LLSF, LLSF-DL and LLSF-BR: We run these methods 5 times with different settings of α, β and γ, and the 5 partitions of training and testing parts of rcv1(subset1) data are fixed. The performance is evaluated in terms of F_1, *Macro* F_1 and *Micro* F_1, and the average results are shown in Fig. 9.

For results of LLSF (Fig. 9(a) to 9(f)) and LLSF-BR (Fig. 9(m) to 9(r)). Given β, the performance is first improved and then declines quickly with the increase of α. The label correlations can be modeled well with an appropriate value of α. Given α, with the increase of β, the performance is improved slightly and then declines dramatically. If β is too small, LLSF can not filter out irrelevant features to each class label, so the performance is relative low. While irrelevant features will be filtered out with the increasing of β so that the performance is improved accordingly. However, the performance declines dramatically when β is too large. Because the *label-specific features* are also filtered out.

For LLSF-DL, β and γ are set to be the same value. The results are shown from Fig. 9(g) to 9(l). Given β and γ, its performance is improved slightly and then degraded with the increase of α; Given α, the result of LLSF-DL is improved slightly with the increase of β and γ, and then declines dramatically when β and γ are too large. If γ is too small, each label is dependent on all the labels, which is equivalent to *stacking*. When γ is too large, we may learn a very sparse dependent structure, so the performance declines quickly. Comparing to LLSF, the performance of LLSF-DL is more stable.

With a fixed setting of α, β, and γ, the performance of LLSF and LLSF-DL is improved slightly and then declines with the increase of ρ, and the best results are obtained at $\rho = 0.1$ or $\rho = 1$ in most cases. Detailed experimental results are provided in the supplementary file.

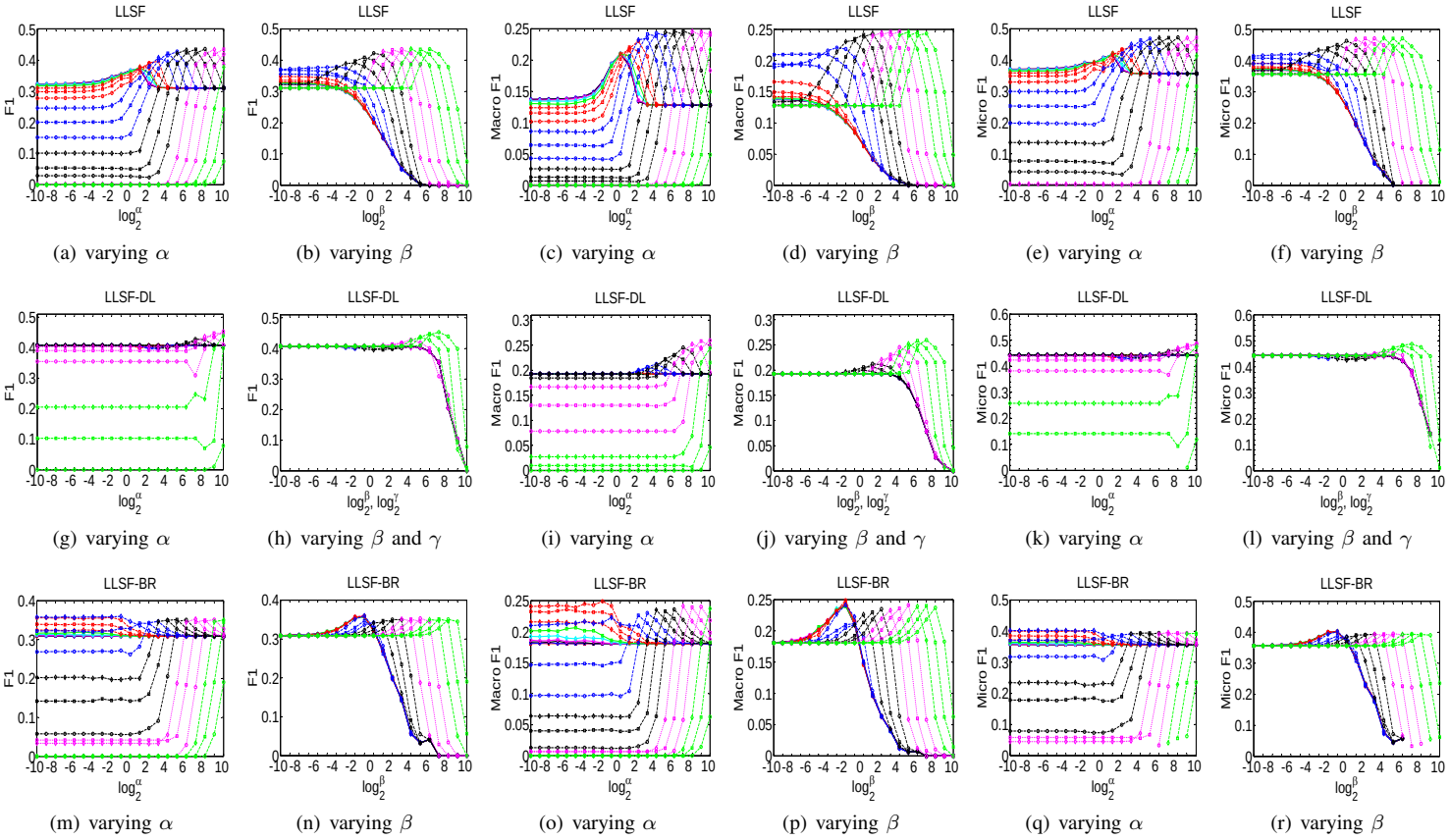


Fig. 9. Parameter sensitivity analysis of LLSF, LLSF-DL and LLSF-BR on rcv1(subset1). The legend is the same as the one shown in Fig. 8.

The performance of our proposed methods are sensitive to the parameters. To use our proposed methods, we suggest that α should be set greater than or equal to β and γ, especially for large scale data sets. More results of parameter sensitivity analysis on other data sets are provided in the supplementary file. In real applications, an appropriate setting of parameters can be searched by cross validation in terms of one evaluation metric or several evaluation metrics simultaneously.

5 Application to Large-Scale Data Set

In this section, we apply the proposed methods to relative large-scale multi-label data. We compare them with three baseline methods: kNN (k is searched in $\{7, 8, ...,11\}$), RidgeReg (Ridge Regression, the regularization parameter is tuned in $\{2^{-10}, 2^{-9}, ..., 2^{10}\}$), and LibLinear (L1R_LR: Logistic Regression with ℓ_1 Norm, C is tuned in $\{10^{-4}, 10^{-3}, ..., 10^{4}\}$). The experiment is conducted on three relative large-scale multi-label data sets, i.e. bookmarks, imdb and nus-wide (see Table 1). Friedman and Nemenyi tests [57] are employed to conduct performance analysis among the comparing algorithms, and the detailed results are provided in the supplementary file. The experimental results show that our proposed methods achieve comparable performance against other three comparing algorithms in terms of each evaluation metric.

6 Conclusion

In this paper, an extension to our preliminary version [23] is presented which learns *label-specific features* and *class-dependent labels* for multi-label classification by modeling *high-order* label correlations in a *sparse stacking* way. The proposed methods LLSF and LLSF-DL can be utilized as a feature selection method for multi-label learning. The learned *label-specific features* for each label can be applied as input to existing multi-label classification algorithms comprising a number of binary classifiers. Comprehensive comparisons with several well-established multi-label classification algorithms over fifteen multi-label benchmark data sets manifest the competitive performance of our methods. Moreover, the performance of LLSF is improved after modeling the *high-order* label correlations. In the future, we will apply our methods to multi-target regression.

References

[1] A. K. McCallum, "Multi-label text classification with a mixture model trained by em," in *AAAI'99 Workshop on Text Learn.*, 1999.

[2] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1338–1351, 2006.

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

[4] F.-M. Sun, J.-H. Tang, H.-J. Li, G.-J. Qi, and T. S. Huang, "Multi-label image categorization with sparse factor representation," *IEEE Trans. Image Process*, vol. 23, no. 3, pp. 1028–1037, 2014.

[5] F. Kang, R. Jin, and R. Sukthankar, "Correlated label propagation with application to multi-label learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2006, pp. 1719–1726.

[6] G.-J. Qi, X.-S. Hua, Y. Rui, J.-H. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *Proc. ACM Multimedia*, 2007, pp. 17–26.

[7] X. Wang and G. Sukthankar, "Multi-label relational neighbor classification using social context features," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 464–472.

[8] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music into emotions," in *Int. Soci. Music Inf. Retri.*, 2008, pp. 325–330.

[9] B. Wu, E.-H. Zhong, A. Horner, and Q. Yang, "Music emotion recognition by multi-label multi-layer multi-instance multi-view learning," in *Proc. ACM Multimedia*, 2014, pp. 117–126.

[10] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," *Data Mining Knowl. Discov. Handbook, O. Maimon, L. Rokach (Ed.), Springer, 2nd edition*, 2010.

[11] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, 2014.

[12] S.-W. Ji, L. Tang, S.-P. Yu, and J.-P. Ye, "Extracting shared subspace for multi-label classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 381–389.

[13] Y. Zhang and Z.-H. Zhou, "Multilabel dimensionality reduction via dependence maximization," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 3, pp. 1–21, 2010.

[14] J. Langford, T. Zhang, D. J. Hsu, and S. M. Kakade, "Multi-label prediction via compressed sensing," in *Proc. Neural Inf. Process. Syst.*, 2009, pp. 772–780.

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

[16] Y.-N. Chen and H.-T. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1529–1537.

[17] T.-Y. Zhou, D.-C. Tao, and X.-D. Wu, "Compressed labeling on distilled labelsets for multi-label learning," *Mach. Learn.*, vol. 88, no. 1-2, pp. 69–126, 2012.

[18] Z.-J. Lin, G.-G. Ding, M.-Q. Hu, and J.-M. Wang, "Multi-label classification via feature-aware implicit label space encoding," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 325–333.

[19] M.-L. Zhang and L. Wu, "Lift: Multi-label learning with label-specific features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 107–120, 2015.

[20] A. Alalga, K. Benabdeslem, and N. Taleb, "Soft-constrained laplacian score for semi-supervised multi-label feature selection," *Knowl. Inf. Syst.*, pp. 1–24, 2015.

[21] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar, "A dirty model for multi-task learning," in *Proc. Neural Inf. Process. Syst.*, 2010, pp. 964–972.

[22] S. Kim, K. Sohn, and E. P. Xing, "A multivariate regression approach to association analysis of a quantitative trait network," *Bioinformatics*, vol. 25, no. 12, pp. 204–212, 2009.

[23] J. Huang, G.-R. Li, Q.-M. Huang, and X.-D. Wu, "Learning label specific features for multi-label classification," in *Proc. IEEE Int. Conf. Data Mining*, 2015, pp. 181–190.

[24] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Proc. Eur. Conf. Mach. Learn.*, 2009, pp. 254–269.

[25] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga, "Bayesian chain classifiers for multidimensional classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 2192–2197.

[26] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 999–1008.

[27] 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 Recognit.*, vol. 47, no. 3, pp. 1494 – 1508, 2014.

[28] K. Abhishek, V. Shankar, M. A. Krishna, and E. Charles, "Beam search algorithms for multilabel learning," *Mach. Learn.*, vol. 92, no. 1, pp. 65–89, 2013.

[29] J. Read, L. Martino, and D. Luengo, "Efficient monte carlo methods for multi-dimensional learning with classifier chains," *Pattern Recognit.*, vol. 47, no. 3, pp. 1535 – 1546, 2014.

[30] G. Tsoumakas, I. Katakis, and L. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 1079–1089, 2011.

[31] F. Briggs, X. Fern, and R. Raich, "Context-aware miml instance annotation: exploiting label correlations with classifier chains," *Knowl. Inf. Syst.*, vol. 43, no. 1, pp. 53–79, 2015.

[32] A. Alali and M. Kubat, "Prudent: A pruned and confident stacking approach for multi-label classification," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2480–2493, 2015.

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

[34] K. Dembczyński, W. Cheng, and E. Hüllermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1609–1614.

[35] W. Bi and J. Kwok, "Bayes-optimal hierarchical multilabel classification," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 2907–2918, 2015.

[36] S.-J. Huang and Z.-H. Zhou, "Multi-label learning by exploiting label correlations locally," in *Proc. AAAI Conf. Artif. Intell.*, 2012.

[37] J. Huang, G.-R. Li, S.-H. Wang, W.-G. Zhang, and Q.-M. Huang, "Group sensitive classifier chains for multi-label classification," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2015, pp. 1–6.

[38] R. Prati, G. Batista, and D. Silva, "Class imbalance revisited: a new experimental setup to assess the performance of treatment methods," *Knowl. Inf. Syst.*, vol. 45, no. 1, pp. 247–270, 2015.

[39] E. S. Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 1583–1588.

[40] M. A. Tahir, J. Kittler, and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," *Pattern Recognit.*, vol. 45, no. 10, pp. 3738–3750, 2012.

[41] M.-L. Zhang, Y.-K. Li, and X.-Y. Liu, "Towards class-imbalance aware multi-label learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015.

[42] X. Kong and P. Yu, "gmlc: a multi-label feature selection framework for graph classification," *Knowl. Inf. Syst.*, vol. 31, no. 2, pp. 281–305, 2012.

[43] J.-H. Chen, J. Liu, and J.-P. Ye, "Learning incoherent sparse and low-rank patterns from multiple tasks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2010, pp. 1179–1187.

[44] J.-H. Chen, J.-Y. Zhou, and J.-P. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 42–50.

[45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[46] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[47] E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, and I. Vlahavas, "Multi-target regression via input space expansion: treating targets as inputs," *Mach. Learn.*, vol. 104, no. 1, pp. 55–98, 2016.

[48] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan, "Matching words and pictures," *J. Mach. Learn. Res*, vol. 3, pp. 1107–1135, 2003.

[49] P. Duygulu, K. Barnard, J. F. G. de Freitas, and D. A. Forsyth, *Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary*. Springer Berlin Heidelberg, 2002, pp. 97–112.

[50] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1:a new benchmark collection for text categorization research," *J. Mach. Learn. Res*, vol. 5, pp. 361–397, 2004.

[51] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas, *Protein Classification with Multiple Algorithms*. Springer Berlin Heidelberg, 2005, pp. 448–456.

[52] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, 2008.

[53] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proc. ACM Int. Conf. Image and Video Retrieval*, 2009, pp. 48:1–48:9.

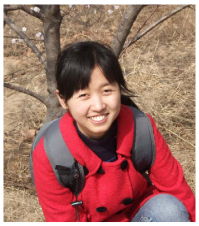
[54] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467–476, 2008.

[55] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "MEKA: A multi-label/multi-target extension to Weka," *J. Mach. Learn. Res*, vol. 17, no. 21, pp. 1–5, 2016.

[56] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011.

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

Jun Huang received the M.S. degree in computer science from Anhui University of Technology, Ma'anshan, China, in 2011. Now, he is a Ph.D. student in School of Computer and Control Engineering, University of the Chinese Academy of Sciences (UCAS). Before joining UCAS, he was a lecturer with Anhui University of Technology. His research interests include machine learning and data mining.



Guorong Li received her B.S. degree in computer science from Renmin University of China, in 2006 and Ph.D. degree in computer science from the Graduate University of the Chinese Academy of Sciences in 2012. Now, she is an associate professor within the University of Chinese Academy of Sciences. Her research interests include object tracking, pattern recognition, cross-media analysis and multi-label learning.

Qingming Huang is currently a professor and deputy dean in the School of Computer and Control Engineering, University of Chinese Academy of Sciences (UCAS). His research interests include multimedia computing, image/video processing, pattern recognition and computer vision. He has published more than 300 academic papers in international journals such as IEEE Transactions on Image Processing, IEEE Transactions on Multimedia, IEEE Transactions on CSVT, and at top level international conferences including ACM Multimedia, ICCV, CVPR, VLDB, IJCAI, etc.

Xindong Wu received the bachelors and masters degrees in computer science from the Hefei University of Technology, China, and the PhD degree in artificial intelligence from the University of Edinburgh, United Kingdom. He is a Yangtze River Scholar in the School of Computer Science and Information Engineering at the Hefei University of Technology, China, and a professor of computer science at the University of Vermont. His research interests include data mining, knowledge-based systems, and Web information exploration. He is the Steering Committee chair of the IEEE International Conference on Data Mining (ICDM), the editor-in-chief of Knowledge and Information Systems (KAIS, by Springer), and a series editor of the Springer Book Series on Advanced Information and Knowledge Processing (AIKP). He was the editor-in-chief of the IEEE Transactions on Knowledge and Data Engineering (TKDE, by the IEEE Computer Society) between 2005 and 2008. He served as a program committee chair/cochair for ICDM 03 (the 2003 IEEE International Conference on Data Mining), KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), and CIKM 2010 (the 19th ACM Conference on Information and Knowledge Management). He is Fellow of the IEEE and AAAS.