IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 13, 2020, accepted March 2, 2020, date of publication March 13, 2020, date of current version March 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980551

Label Specific Features-Based Classifier Chains for Multi-Label Classification

WEI WENG[1,2], DA-HAN WANG[1], CHIN-LING CHEN[1,3,4], JUAN WEN[5], AND SHUN-XIANG WU[6]

[1]College of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China
[2]Key Laboratory of Data Science and Intelligence Application, Fujian Province University, Zhangzhou 363000, China
[3]School of Information Engineering, Changchun Sci-Tech University, Changchun 130600, China
[4]Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung 41349, Taiwan
[5]Department of Statistics, School of Economics, Xiamen University, Xiamen 361005, China
[6]Department of Automation, Xiamen University, Xiamen 361005, China

Corresponding authors: Chin-Ling Chen (clc@mail.cyut.edu.tw) and Juan Wen (wenjuan@xmu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61773325, Grant 61801413, and Grant 61871464, in part by the Natural Science Foundation of the Fujian Province under Grant 2018J01572 and Grant 2018J01574, in part by the Key Laboratory of Data Science and Intelligence Application, Fujian Province University, under Grant D1803, and in part by the Social Science Foundation of Fujian Province under Grant JAS150057.

ABSTRACT Multi-label classification tackles the problems in which each instance is associated with multiple labels. Due to the interdependence among labels, exploiting label correlations is the main means to enhance the performances of classifiers and a variety of corresponding multi-label algorithms have been proposed. Among those algorithms Classifier Chains (CC) is one of the most effective methods. It induces binary classifiers for each label, and these classifiers are linked in a chain. In the chain, the labels predicted by previous classifiers are used as additional features for the current classifier. The original CC has two shortcomings which potentially decrease classification performances: random label ordering, noise in original and additional features. To deal with these problems, we propose a novel and effective algorithm named LSF-CC, i.e. *Label Specific Features based Classifier Chain for multi-label classification.* At first, a feature estimating technique is employed to produce a list of most relevant features and labels for each label. According to these lists, we define a chain to guarantee that the most frequent labels that appear in these lists are top-ranked. Then, label specific features can be selected from the original feature space and label space. Based on these label specific features, corresponding binary classifiers are learned for each label. Experiments on several multi-label data sets from various domains have shown that the proposed method outperforms well-established approaches.

INDEX TERMS Classifier chains, label specific features, multi-label learning.

I. INTRODUCTION

The traditional single-label classification aims to assign a label for each instance from a finite set of labels, including binary and multi-class classification [1]. However, real-world applications usually need to assign multiple labels simultaneously. For example, a large number of documents on the web site often require several topics with the purpose of the improved quality of search and recommendation [2]; automatically assigning keywords to an uploaded web image makes it easy to be searched directly using text-based image retrieval systems [3]; in social network analysis, with the help of multiple labels associated with nodes, accuracy and stability methods of community discovery with linear time complexity can be obtained to reveal the organizational principle and dynamic characteristic of the real network [4]; annotating image-level labels for high-resolution aerial imagery offers a holistic understanding of those images and makes tasks of urban mapping, ecological monitoring and geomorphological analysis more effective at low cost [5]. Multi-label classification is a supervised machine learning framework for these scenarios, which aims at automatically assigning correct labels to unseen instances. In recent years, multi-label classification has become a hot topic in the research area of data mining and machine learning due to its extensive applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Julien Le Kernec.

One popular strategy for multi-label classification is problem transformation [6], which decomposes the original problem into one or more binary classification subproblems. Many algorithms in multi-label learning have followed this strategy, in which binary relevance (BR) [7] is a representative one. BR constructs a subproblem for each label, and the instance associated with the label is viewed as a positive one, otherwise negative. BR has one obvious shortcoming, i.e. it trains binary classifiers for each label independently and completely ignores the dependence relationships among labels. Because of this shortcoming, it usually gives rise to the suboptimal prediction performance when strong label correlation exists. For example, an image tagged with ''desert'' usually is likely to be tagged with ''camel''. Due to the existence of dependency relationships among labels, multi-label classification tends to be more challenging than traditional single-label classification. Exploiting label correlation has become the main impetus in multi-label learning to improve prediction performance [8]–[13]. For example, to incorporate label correlations in BR, the Classifier Chains (CC) [9] was proposed. CC links these binary classifiers in a given sequential order, such that each classifier incorporates the labels predicted by the previous classifiers as additional features. Despite its simplicity, a comprehensive recent empirical study demonstrated that CC is among the top best-performing algorithms [14].

Except for the challenge of label correlations, the high-dimensional feature space is another challenge in multi-label learning [2], [15], [16]. As we know, the noise and irrelevant features within the data also put an obstacle to a given learning task, such as computational burden, over-fitting, and poor performance. To solve these problems, a lot of feature reduction based approaches have been presented, and most of them construct an identical feature subset for all labels [17]. Recently, Zhang and Wu [18] discussed that each label in multi-label problems is supposed to determine by some specific features of its own, so existing classifiers learning from the identical feature set for all labels might be suboptimal. From then on, label specific features attracted great interest in multi-label learning [11], [19]–[22]. According to the type of extracting label specific features, existing algorithms can be roughly divided into two major categories: label-specific feature extraction and label-specific feature selection. label-specific feature extraction converts high-dimensional feature space into low-dimensional feature space for each label through transforming or mapping. Because the translated features are completely different from the original features, it is difficult to understand the relationship between them. LIFT [18] is one of the representative algorithms of feature extraction. It exploits the label specific features by conducting clustering analysis on positive and negative instances respectively, and the new features are represented by distances between the original instances and the centers of clusters. Label-specific feature selection attempts to eliminate as many as possible redundant features from the original feature space and preserve discriminative features for each label, which can preserve the original meaning of features. LLSF [23] is one of the representative algorithms of feature selection. It exploits label specific features through linear regression with ℓ_1 norm modeling the discriminant and sparsity of them.

Motivated by the aforementioned works, we propose a novel classifier chain method named LSF-CC, i.e. Label Specific Features based Classifier Chains for multi-label classification. LSF-CC follows the common framework of the classifier chain method, and constructs binary classifiers in a predetermined ordering, in which each classifier uses the predictions of previous classifiers in the chain as additional features. Because the wrong predictions would propagate along the chain, and an inadequate label ordering can potentially decrease the performance of classifier chain methods. Another important issue of the classifier chain method is that the features, no matter the original feature or the additional features, usually include redundant and irrelevant features, which may bring disadvantages to learning algorithms. To deal with those problems, LSF-CC optimizes the label ordering in a chain of classifiers according to label correlations and selects label specific features from original feature space and label space. Then, it learns the corresponding binary classifiers with them. Comparison with CC and other state-of-the-art manifests the efficiency of our proposed method. The key contributions of our method are summarized as follows:

1) Our proposed algorithm LSF-CC figures out the two shortcomings of traditional classifier chain based algorithms which can potentially decrease their performance: the random label ordering and the noises in original and additional features.
2) Our method considers label correlations from a global perspective. For each label, the relations between it and all outputting labels are measured.
3) Existing feature selection algorithms usually extract features from feature space, while the proposed algorithm can select label specific features for each label from feature and label space.

The rest of this article is organized as follows. Section II briefly reviews the related work on multi-label learning. In Section III, we describe the detail of our proposed algorithm LSF-CC. Parameter settings, data sets and experimental results are shown in Section IV. Finally, Section V concludes this paper.

II. RELATED WORKS

Early multi-label classification originated from the investigation of multi-label text categorization techniques [24]. At that time, what is driving these studies is the need for efficient retrieval of large numbers of documents on the Internet. Nowadays, with the continuous development and in-depth application of information collection, data transmission and web services in all walks of life, a large amount of multi-label

data has been accumulated, and the researches on multi-label classifications have been gradually extended to image and video annotation, gene function, music emotion classification and other fields. In recent years, a lot of multi-label classification algorithms have been presented, which can be divided into two main categories: problem transformation and algorithm transformation [6].

Problem transformation methods decompose the multi-label problem into one or more single-label subproblems, then they can directly figure out via traditional classification models. To do it, problem transformation methods mainly depend on three strategies: binary relevance, label powerset and pairwise. Binary relevance methods learn a binary classifier for each label. BR [7] is one of the representative binary relevance methods. It constructs binary classifiers independently, in which the label correlations are not considered at all. To incorporate label correlation into BR, Godbole and Sarawagi [25] proposed stacking BR model, which contains two-layer of BR. In the meta-level, all the predicts of the base label classifiers are used as additional features for learning the final binary classifiers. Because label correlations usually exist among part rather than all of the labels, this type of stacking BR model usually introduces noises or redundant features into learning. A few works try to select a part of the outputs of the base level BR as additional features for meta-level learning, such as BR+ [26], SMBPO [11]. Read *et al.* proposed classifier chains (CC) [9] model for multi-label classification, which constructs a chain of binary classifiers, each for a label. CC learns these classifiers one by one, and in the chain each classifier is learned using the original features augmented with all labels associated with the previous classifiers in the chain. It works by a random label sequence, so its performance is seriously constrained by the choice of label ordering. If the previous predictions are wrong, the errors will propagate in the next steps. In order to mitigate the error propagation introduced by random ordering, some methods resort to an ensemble of classifier chains [9], other works turn to optimize the label ordering [27]. Label powerset methods treat related labels of each instance as an atomic label. Then, the multi-label problem is transformed to a single-label multi-class problem, such as RAkEL [28] and EPS [29]. Label powerset methods overcome the label independence problem, but suffer from the huge label combinations when the label set is large and labeling is very variable. Pairwise methods decompose the multi-label problem with q labels into $q(q-1)/2$ binary classification subproblems, in which each subproblem is responsible for a pair of labels. Then, the predictions are combined into a ranking based on the ensemble of those binary classifiers, such as PRC [30] and PRC [31]. Obviously, pairwise methods are greatly affected by the number of labels, and when there are too many labels, those methods are too complex to be practical.

Problem transformation methods are independent from specific classification algorithms, while algorithm transformation methods adapt existing single-label algorithms for the purpose of multi-label classification. Much of the literature is focussed on modifications to decision trees [32], k nearest neighbors [34], neural network [35] and support vector machine [36]. ML-C4.5 [37] builds a decision tree through a top-down approach, with the root containing all the training samples. For a non-leaf node in the tree, each feature is examined one by one to find an appropriate dividing point, so that the data of the node can be divided to obtain the maximum information gains. Clus-HMC [32] employed a decision tree for hierarchical multi-label prediction. It divides the sample set in the current node into several disjoint subsets with the principle of minimizing intra-cluster variance. In Clus-HMC, the intra-cluster variance of a node is defined as the distances between each label vector of instance and the average label vector. Bp-MLL [35] firstly employs a neural network for multi-label classification, which designs an optimized objective function to discriminate label pairs between the relevant labels and the irrelevant labels. RankSVM [36] proposed a multi-label classification algorithm based on a large margin ranking system that shares common properties with SVMs.

III. THE PROPOSED ALGORITHM

Let $\boldsymbol{x}_i \in \mathcal{X} = \mathcal{R}^d$ denotes the instance with d features, $\mathcal{Y} = \{y_1, y_2, \ldots, y_q\}$ are the label set. The multi-label training set $D = \{(x_i, Y_i) | 1 \leq i \leq n\}$ consists of n instances, where $Y_i \subseteq \mathcal{Y}$ is the labels associated with x_i. For convenience, Y_i is often represented as a logic vector c_i. If $y_j \in Y_i$, then $c_{ij} = 1$; otherwise $c_{ij} = 0$. Therefore, D can be represented as the combination of input matrix $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T$ and output matrix $Y = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n]^T$. The task of multi-label learning is to define a predictor $h : \mathcal{X} \to \{1, -1\}^q$ from D. If we learn classifiers for each label, then $h = [h_1, h_2, \ldots, h_q]$. And, for the label y_i, its predictor $h_i : \mathcal{X} \to \{1, 0\}$.

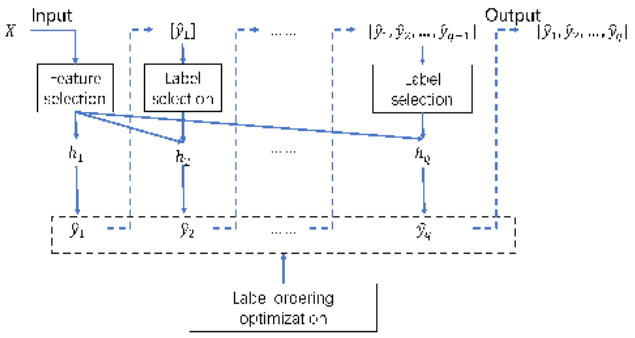


FIGURE 1. The framework of LSF-CC.

Our proposed algorithm, LSF-CC, is shown in Figure 1, which includes three steps: label ordering optimization, label selection and feature selection. In the first step, we establish an optimized sequence of labels according to label correlations, so that the labels at the beginning of the sequence are

as dependent as possible on the labels that follow them. For the current label in the sequence, its previous labels may not all be related to it, that is, not all those labels are suitable to be the new features for the label. Therefore, in the step of label selection, for a label in the sequence, we select the most related labels from previous ones as its new feature. In the step of feature selection, the label-specific features are achieved by removing irrelevant and redundant features from the original feature space in a filtered manner. Then, the corresponding binary classifiers are learned based on those label-specific features coming from feature space and label space.

A. LABEL ORDER OPTIMIZATION

The biggest problem of the original CC algorithm lies in the unstable operation effect caused by the random label ordering for training, which is difficult to be applied to practical problems. Therefore, it is necessary to select an adequate label ordering and train the corresponding binary classifiers according to it, so as to achieve the goal of improving the performance of the classifier chain algorithm. In order to optimize the ordering of labels, first the correlations among labels are estimated. According to those correlations, we select the most related label subset R_i for each label y_i. Then, the number of the appearance of each label is accounted based on those subsets $R_1, R_2, \ldots, R_q$. The more times a label appears, the more other labels are correlated with it. For those frequent labels, they should be predicted as early as possible, so that their predicting results can be used as additional features for correlated labels. In other words, those labels should be ranked first in the label ordering.

In order to estimate label correlations, here the well-known feature estimating method ReliefF [38] is used. ReliefF considers that good features should differentiate between instances from different classes and should have the same value for instances from the same class. For a given instance $\boldsymbol{x}_i$, ReliefF searches its k-nearest neighbors from the same and different classes respectively. Then, ReliefF penalizes every feature which has different values between $\boldsymbol{x}_i$ and instances from the same class and rewards ones that have different values between $\boldsymbol{x}_i$ and instances from different classes. ReliefF can be applied on discrete or continuous features. Furthermore, ReliefF can deal with noisy, incomplete and multi-class data sets. Because of those characteristics, ReliefF was shown to be very efficient in estimating features. Recently ReliefF is used in multi-label learning for feature selection [39], [40]. Existing methods usually utilize ReliefF to estimate features. In our algorithm labels are used as additional features to construct classification models. So, we give an attempt to estimate labels by ReliefF. Similar to features, good labels should differentiate between logic vectors from different classes and should have the same value for logic vectors from the same class. Formally, for a label $y_l (1 \leq l \leq q)$, function $diff(y_l, \boldsymbol{c}_i, \boldsymbol{c}_j)$ calculates the difference between the values of y_l for two logic vectors $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$. As the value of y_l is discrete, the function is defined in Eq.(1):

$$diff(y_l, \boldsymbol{c}_i, \boldsymbol{c}_j) = \begin{cases} 0 & c_{il} = c_{jl} \\ 1 & c_{il} \neq c_{jl} \end{cases} \tag{1}$$

We let a vector $\boldsymbol{w}_l = [w_{l1}, w_{l2}, \ldots, w_{lq}]^T$ record the estimating values of all labels in terms of label y_l. Each element $w_{le} (1 \leq e \leq q)$ ranges from -1 to 1, and it is initially set as 0. The larger value of w_{le} indicates the label y_e is more important for label y_l. w_{le} is repeatedly calculated for z times. For each time, we randomly select a label vector $\boldsymbol{c}_i$ and search for its two $k-$nearest neighbor set: one from the same class, denoted by H_i, and the other from a different class, denoted by M_i. Then w_{le} is updated by Eq.(2).

$$w_{le} = w_{le} - \sum_{j=1}^{k} diff(y_e, \boldsymbol{c}_i, H_i(j))/(zk) + \sum_{j=1}^{k} diff(y_e, \boldsymbol{c}_i, M_i(j))/(zk) \tag{2}$$

Algorithm 1 presents the whole process of label ordering optimization. In Algorithm 1, Step 1 estimates the qualities of all labels according to the ReliefF method. Steps 2-6 select the p most relevant labels for each label and save them in *lacorelaiton*. Step 7 counts the times of appearance of each label y_i in *lacorelaiton*$(i, :)$. Step 8 sorts the statistical results in descending order and save corresponding labels in LI. Step 9 initializes the *list* with the first label in LI, as most labels relate to it. Steps 10-21 continue to select appropriate labels to input *list* in sequence. The candidate label needs to be judged with two conditions: 1) it appears as many times as possible in *lacorelaiton*, which means that there are as many subsequent labels depending on them as possible; 2) existing labels in the current label ordering should be as relevant as possible with it. These two conditions are intended to ensure that the previous labels in the ordering are as dependent as possible by the subsequence labels, so that they are suitable to be the additional features of the current label. To guarantee the priority of the label which satisfies condition (1), the loop in steps 12-15 traverse $L1$ from front to end. Note that the loop may not find the label satisfies the condition (2). Step 11 initializes *candidate*1 and *candicate*2, where variable *candidate*1 records the label that satisfies the condition (1) and the variable *candidate*2 records the label that satisfies the conditions (1) and (2). Step 13 searches for the label that satisfies the condition (1). Step 14 searches for the label that satisfies the conditions (1) and (2). If *candidate*2 is successfully found, Step 17 assigns *candidate*2 to *list*(i), otherwise, *list*(i) is *candidate*1.

Example 1: In order to demonstrate Algorithm 1 more clearly, hear the data set Emotions (described in Section IV) is used to describe the whole process of label ordering optimization. Emotions had six labels $y_1, y_2, \ldots, y_6$. We randomly select four-fifths of its instances for running Algorithm 1. In terms of performing Step 1, the results of W are shown as (3), at the bottom of the next page.

Algorithm 1 Label Ordering Optimization Based on Label Correlations

Input:
Y: the output matrix;
p: the cardinality of the related label subsets
Output:
lacorelaiton: the most related label subsets;
list: the label ordering;

1: estimate all labels for y_i and save them in $W[i, :]$;
2: **for** each label y_i **do**
3: sort $W[i, :]$ in descending order;
4: save the corresponding labels of previous p elements of $W[i, :]$ in $lacorelaiton[i, :]$;
5: $lacorelaiton[i, ;] \leftarrow lacorelaiton[i, :] - y_i$;
6: **end for**
7: count the frequency of each label in *lacorelaiton* and save it in *occnum*;
8: sort *occnum* in descending order, and save the corresponding labels in LI;
9: $list(1) \leftarrow LI(1)$;
10: **for** $i = 2 : q$ **do**
11: $candidate1 \leftarrow -1$; $candidate2 \leftarrow -1$;
12: **for** $j = 2 : q$ **do**
13: search for j until $LI(j) \notin list$, then set $candidate1 \leftarrow j$;
14: Search for j until $LI(j) \notin list$ and $list \cap lacorelaiton(LI(j), :) \neq \emptyset$, then set $candidate2 \leftarrow j$;
15: **end for**
16: **if** candidate2 == -1 **then**
17: $list(i) \leftarrow LI(candidate1)$
18: **else**
19: $list(i) \leftarrow LI(candidate2)$
20: **end if**
21: **end for**

The value of p is set to be 5. after running Steps 2-6, we can get the *lacorelaiton*:

$$lacorelaiton = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 1 & 4 & 5 & 6 \\ 1 & 3 & 5 & 6 \\ 1 & 2 & 3 & 6 \\ 2 & 3 & 4 & 5 \end{bmatrix} \tag{4}$$

According to Step 7, $occnum = [3, 3, 5, 4, 5, 4]$. It denotes the times of occurrence of $y_1, y_2, \ldots, y_6$ in *lacorelaiton* respectively. Then, $LI = [3, 5, 4, 6, 1, 2]$ in the Step 8. It illustrates among all labels y_3 is the most frequent one, then y_5 follows, and so on. According to Step 9, $list(1) = y_3$. Steps 10-26 determine the order of the subsequence five labels. When $i = 2$, we determine the second label of the *list*. First, $LI(2)$, i.e. y_5 is to be considered as a candidate. Because $y_3 \in lacorelaiton(5, :)$, that is, the conditions in Step 16 hold, $list(2) = y_3$. By that analogy, the final result of the *list* is $y_3, y_5, y_4, y_6, y_1, y_2$. It is the optimized label ordering.

Algorithm 2 Label Selection

Input:
lacorelaiton: the most related label subsets;
list: the label ordering;

Output:
lsl: the selected label subset for each label;

1: $prelb = []$;
2: **for** $i = 1 : length(list)$ do **do**
3: $curlb = list(i)$;
4: $lsl[curlb, :] = []$;
5: **for** $in = 1 : length(prelb)$ do **do**
6: **if** $ismenber(prelb(in), lacorelaiton[curlb, :])$ **then**
7: $lsl[curlb, :] = [lsl[curlb, :], prelb(in)]$;
8: **end if**
9: **end for**
10: $prelb = [prelb, list(i)]$;
11: **end for**

B. LABEL SELECTION

As mentioned in Section I, for each label the original CC employs all its previous labels as additional features. Because there may not exist correlations between the current label and all its previous labels, it will degenerate the performances of classifiers. In section III-A, we have presented how to compute the label correlations and the optimal label ordering. With the result of Algorithm 1, for each label the most related label subset can be selected. Note that *lacorelaiton* records the corresponding related labels for each label, it can not be treated as additional features directly, as the labels in *lacorelaiton* may not locate in front of the current label. So, for each label y_i, its selected label subset consists of those labels which locate in $lacorelaiton[i, :]$ and in front of it in the *list*. The process of feature selection is shown in Algorithm 2.

$$W = \begin{bmatrix} 1 & 0.0212 & 0.1948 & 0.2200 & 0.0440 & 0.0192 \\ 0.0008 & 1 & 0.0192 & 0.0312 & 0.2600 & 0.0516 \\ 0.1104 & -0.0020 & 1 & 0.0152 & 0.0360 & 0.2172 \\ 0.2388 & 0.0144 & 0.1788 & 1 & 0.0604 & 0.2012 \\ 0.0292 & 0.2752 & 0.0580 & -0.0016 & 1 & 0.0640 \\ -0.0068 & 0.0988 & 0.2672 & 0.1576 & 0.0500 & 1 \end{bmatrix} \tag{3}$$

In Algorithm 2, *prelb* denotes the previous labels of the label *curlb*, and the selected label subset is saved in $lsl[curlb, :]$. Steps 5-9 present the process of label selection for the label *curlb*, in which the labels in *prelb* are checked one by one whether they are related with *curlb*. The related labels are saved in $lsl[curlb, :]$.

Example 2: With the results *lacorelaiton* and *list* of example 1, Algorithm 2 is ran and the process of label selection for each label is shown in Table 1. As shown in Table 1, the selected label set of label y_3 is none, as it is the first label in the label ordering *list* and need none of labels as additional features. The selected label subset for y_5 is $\{y_3\}$ and the selected label subset for y_4 is $\{3, 5\}$, and so on. Note that for label y_1, its selected label set is $\{y_3, y_5, y_4\}$, which is a subset of its previous labels in the label ordering *list*.

TABLE 1. An example of label selection.

Variables	Values					
i	1	2	3	4	5	6
$prelb$	$\{\}$	$\{3\}$	$\{3,5\}$	$\{3,5,4\}$	$\{3,5,4,6\}$	$\{3,5,4,6,1\}$
$curlb$	y_3	y_5	y_4	y_6	y_1	y_2
$lsl[curlb,:]$	$\{\}$	$\{3\}$	$\{3,5\}$	$\{3,5,4\}$	$\{3,5,4\}$	$\{3,5,4,6\}$

C. FEATURE SELECTION

Different from the original CC, considering the original feature space may include redundant or irrelevant features, in LSF-CC we select the appropriate feature subset for each label to train. Here the ReliefF is employed to estimate the quality of features, then the feature subset is selected according to a ratio parameter r, which identifies the percentage of the selected features. The process of feature selection is show in Algorithm 3.

Algorithm 3 Feature Selection

Input:
$\mathcal{D}$: the training set;
r: the ratio of the selected features;

Output:
lsf: the selected features for each label;

1: estimate all labels for y_i and save them in $W[i, :]$;
2: **for** each label y_i **do**
3: sort $W[i, :]$ in descending order;
4: select corresponding feature subset which locate in front of $W[i, :]$ from original feature space according to r, and save those features in $lsf[i, :]$;
5: **end for**

IV. EXPERIMENTS

A. EVALUATION METRICS

The outputs of a test instance in multi-label learning involve multiple labels simultaneously, which could be partially correct, fully wrong or fully correct. This makes the traditional single-label classification evaluation metrics, such as recall, precision and **F**-measure are not suitable for evaluating the performance of multi-label algorithms [41]. Therefore, a variety of evaluation metrics for multi-label learning are proposed, in which we use five widely-used evaluation metrics to verify the performance in our experiments [6], [41]. Those metrics include Hamming Loss, Micro-F1, Macro-F1 and Exact-Match. We let test data set $\mathcal{T} = \{(x_i, Y_i)|1 \leq i \leq m\}$, where Y_i is the ground truth labels and $\hat{Y}_i$ is its predicted labels. These performance evaluation metrics are defined as follows.

1) Hamming Loss

Hamming Loss evaluates how many times the labels are misclassified, which returning the mean value across the test set. We compare the pairs between the predictive labels and the ground truth labels, when there is an inconsistency, i.e. one label is "1" but the other is "0", it shows the label is misclassified. Eq.(5) shows the expression of the Hamming Loss.

$$Hamming\ Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{q}\sum_{j=1}^{q}[\![Y_{ij} \neq \hat{Y}_{ij}]\!], \tag{5}$$

here $[\![\cdot]\!]$ is an indicator function. If the logic expression in it is true, it returns 1; otherwise, it returns 0.

2) Micro-F1

Micro-F1 evaluates the F-measure averaging on the prediction matrix. It is formally described as

$$Micro - F1 = \frac{2\sum_{j=1}^{q}\sum_{i=1}^{m} Y_{ij}\hat{Y}_{ij}}{\sum_{j=1}^{q}\sum_{i=1}^{m} Y_{ij} + \sum_{j=1}^{q}\sum_{i=1}^{m}\hat{Y}_{ij}}. \tag{6}$$

3) Macro-F1

Macro-F1 evaluates the F-measure averaging on each label, as represented in Eq.(7).

$$Macro - F1 = \frac{1}{q}\sum_{j=1}^{q}\frac{2\sum_{i=1}^{m} Y_{ij}\hat{Y}_{ij}}{\sum_{i=1}^{m} Y_{ij} + \sum_{i=1}^{m}\hat{Y}_{ij}}. \tag{7}$$

4) Exact-Match

Exact-Match evaluates how many times the ground truth labels and the predicted labels are exactly matched. It is defined as follows:

$$Exact - Match = \frac{1}{m}\sum_{i=1}^{m}[\![Y_i = \hat{Y}_i]\!]. \tag{8}$$

The domain of values of those evaluation metrics discussed above all vary between [0,1]. For Hamming Loss, the smaller the values the better the performance. For Macro-F1, Micro-F1 and Exact-Match, the larger the values the better the performance.

B. COMPARISON METHODS

As discussed in Section III, the proposed LSF-CC has two characteristics. One is that the label ordering is optimal. The other is that for each label its classifier is learned based

on label specific features which consist of selected features extracting from the original feature and label space. In order to obtain insight or deeper understanding of LSF-CC, we also explore the performance of LSF-CC which extracts label specific features from original feature spaces rather than label space. We denote the original LSF-CC as LSF-CC-PL and its variant as LSF-CC-AL. In these algorithms, the parameters $k = 5$, $m = 500$, and $p = q * 0.5$. r is searched from $\{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$ to obtain the best overall performance.

As for the original algorithm CC, it also can be changed in the same manner. First, the label ordering can be optimized according to the method described in Section III-A. We denote it as CC-LO. Furthermore, according to the method in Algorithm 2, the label subset are selected as additional features for CC. We denote it as CC-LS. As one of representative methods of problem transformation, BR [7] is also compared with our proposed algorithm.

For all the above methods LSF-CC-PL, LSF-CC-AL, CC-LS, CC-LO and BR, they belong to problem transformation methods and need a base classifier for each label. In our experiments, we employ two common classifiers as base classifier: the k-nearest neighbours with $k = 5$ and LIBSVM [33] with linear kernel.

We also compare LSF-CC with three other kinds of algorithms which do not belong to the classifier chain based category: ML-kNN [34], LPLC [42] and LLSF [23]. ML-kNN is a multi-label classifier deriving from the traditional k-nearest neighbor technique and Baysian inference. For each test instance, its k nearest neighbors in the training data are firstly founded. Then, according to the statistical information about the ground truth labels of these neighboring instances, maximum a posteriori principle is utilized to determine which labels are associated with the test instance. LPLC finds the most correlated labels for ground truth labels of each instance in the training stage, then make prediction through estimating with the distribution of each label in the k nearest neighbors and their most correlated local pairwise label correlations. For both ML-kNN and LPLC, the parameter k is set as 10. LLSF can be viewed as a feature selection method that employs linear regression with ℓ_1 norm to model the sparsity of label specific features. In our experiments, its weighting parameters α, β and γ are set to be 0.1, 0.1 and 0.01 respectively, as introduced in the related paper.

The experimental settings and corresponding characteristics for all comparing algorithms are shown in Table 2. Here, ''CC based'' denotes whether the algorithm belongs to Classifier chain based one, while ''LC'', ''LO'', ''FS'' and ''LS'' denote whether the algorithm utilizes label correlations, label ordering, feature selection and label selection for classification.

TABLE 2. The experimental settings and corresponding characteristics for all comparing algorithms.

Method	Parameter	CC based	LC	LO	FS	LS
BR	Non	no	no	no	no	no
CC	Non	yes	yes	no	no	no
CC-LO	$k = 5, m = 500, p = q * 0.5$	yes	yes	yes	no	no
CC-LS	$k = 5, m = 500, p = q * 0.5$	yes	yes	yes	no	yes
LSF-CC-PL	$k = 5, m = 500, p = q * 0.5$	yes	yes	yes	yes	yes
LSF-CC-AL	$k = 5, m = 500, p = q * 0.5$	yes	yes	yes	yes	no
ML-kNN	$k = 10$	no	no	no	no	no
LPLC	$k = 10$	no	yes	no	no	no
LLSF	$\alpha = 0.1, \beta = 0.1, \gamma = 0.01$	no	yes	no	yes	no

TABLE 3. Detailed information on the experimental data set.

Data set	$\mathcal{D}$	$dim(\mathcal{D})$	$L(\mathcal{D})$	Cardinality	Density	Domain	URL
Image	2000	294	5	1.236	0.247	image	URL[1]
Emotions	593	72	6	1.868	0.311	music	URL[2]
Medical	978	1449	45	1.245	0.027	text	URL[2]
Languagelog	1460	1004	75	1.180	0.015	text	URL[3]
Enron	1702	1001	53	3.378	0.064	text	URL[2]
Yeast	2417	103	14	4.237	0.302	biology	URL[2]
Slashdot	3782	1079	22	1.181	0.054	text	URL[2]
Genbase	662	1185	27	1.252	0.046	biology	URL[2]

[1] http://cse.seu.edu.cn/people/zhangml/resources.htm ♯ data
[2] http://mulan.sourceforge.net/datasets.html
[3] http://meka.sourceforge/ ♯ datasets

C. DATASETS

We carried out experiments on eight multi-label benchmark data sets with different types and sizes, which are summarized in Table 3. Here, $\mathcal{D}$, $dim(\mathcal{D})$, $L(\mathcal{D})$ and ''Cardinality'' denote the number of instances, features, labels and the average number of labels over all instances, while ''Density'' is defined as the division of cardinality by the number of labels.

D. EXPERIMENTAL RESULTS AND DISCUSSION

For all the evaluation metrics, we run each algorithm by performing 5-fold cross-validation on the training set and the average predictive performances on test sets are reported. For each evaluation metric, ''↑'' indicates the larger the value the better the performance, while ''↓'' indicates the smaller the value the better the performance. Each result consists of *mean* and *rank*. The best results over each dataset are highlighted in bold type. If two or more algorithms achieve the same performances on one data set for a given evaluation metric, the values of the corresponding *rank* are assigned with the average result of them. To demonstrate the results more clearly, we compute the average rank for each algorithm over all evaluation metrics on a specific data set, which is recoded in the last column for each table.

We conduct two groups of experiments on the eight data sets introduced in Section IV-C. In the first group of experiments, we compare our proposed algorithm with CC-LS, CC-LO and BR, which all belong to problem transformation methods. These algorithms all need a base classifier for each label. Here we employ SVM and 5-NN as the base classifier respectively. In the second group of experiments, we compare the performance of our proposed algorithm with MLkNN, LPLC and LLSF, which all belong to the category of problem transformation. Note that MLkNN and LPLC employ maximum a posteriori principle for classification and LLSF employs regression weights for classification, instead of SVM or 5-NN as the base classifier.

TABLE 4. Results of each comparing algorithm with svm as a base classifier on Image, Emotions, Medical and Languagelog.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Micro-F1 ↑	Macro-F1 (↑)	Exact-Match ↑	Avg. rank
Image	BR	0.1771 3	0.5455 6	0.5415 6	0.3615 6	5.25
	CC	0.1849 4	0.5979 3	0.5941 4	**0.4775 1**	3
	CC-LO	0.1854 5	**0.6021 1**	**0.6027 1**	0.4685 2	**2.25**
	CC-LS	0.1764 2	0.5978 4	0.5951 3	0.4355 5	3.5
	LSF-CC-PL	**0.1747 1**	0.5988 2	0.5960 2	0.4415 4	**2.25**
	LSF-CC-AL	0.1917 6	0.5871 5	0.5867 5	0.4570 3	4.75
Emotions	BR	**0.1967 1**	0.6381 6	0.5899 6	0.2732 6	4.75
	CC	0.2172 6	0.6554 5	0.6302 5	0.3019 2	4.5
	CC-LO	0.2060 4	0.6611 4	0.6498 2	0.2917 4	3.5
	CC-LS	0.2012 2	0.6684 2	0.6437 4	0.2782 5	3.25
	LSF-CC-PL	0.2030 3	**0.6693 1**	**0.6582 1**	0.3018 3	**2**
	LSF-CC-AL	0.2085 5	0.6612 3	0.6481 3	**0.3239 1**	3
Medical	BR	0.0101 4	0.8025 5	0.3546 4	0.6493 6	4.75
	CC	0.0098 2.5	0.8079 3	0.3430 5	**0.6983 1**	2.875
	CC-LO	0.0104 6	0.8010 6	0.3585 2	0.6646 5	4.75
	CC-LS	0.0102 5	0.8057 4	**0.3597 1**	0.6718 4	3.5
	LSF-CC-PL	**0.0097 1**	**0.8116 1**	0.3521 6	0.6810 3	2.75
	LSF-CC-AL	0.0098 2.5	0.8105 2	0.3551 3	0.6830 2	**2.375**
Languagelog	BR	0.1690 5	0.4758 5	**0.2156 1**	0.1541 6	4.25
	CC	0.1698 6	0.4691 6	0.2116 4	0.1452 4	5
	CC-LO	0.1664 3	0.4798 3	0.2138 2	0.1555 3	2.75
	CC-LS	0.1665 4	0.4786 4	0.2130 3	0.1548 5	3.75
	LSF-CC-PL	0.1627 2	0.4847 2	0.2053 6	**0.1712 1.5**	2.875
	LSF-CC-AL	**0.1618 1**	**0.4871 1**	0.2089 5	**0.1712 1.5**	**2.125**

TABLE 5. Results of each comparing algorithm with svm as a base classifier on Enron, Yeast, Slashdot and Genbase.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Micro-F1 ↑	Macro-F1 (↑)	Exact-Match ↑	Avg. rank
Enron	BR	**0.0501 1**	0.4768 6	0.1009 6	0.0916 6	4.75
	CC	0.0525 5	0.4810 5	0.1164 3	0.1111 5	4.5
	CC-LO	0.0526 6	0.4884 4	**0.1258 1**	0.1146 2.5	3.375
	CC-LS	0.0521 4	0.4905 3	0.1247 2	**0.1199 1**	**2.5**
	LSF-CC-PL	0.0503 2	**0.5289 1**	0.1076 5	0.1128 4	3
	LSF-CC-AL	0.0507 3	0.5256 2	0.1077 4	0.1146 2.5	2.875
Yeast	BR	**0.2012 1**	0.6249 6	0.3176 6	0.1340 6	4.75
	CC	0.2181 6	0.6295 5	0.3478 5	**0.2002 1.5**	4.375
	CC-LO	0.2029 4	0.6453 2.5	0.3773 2	**0.2002 1.5**	**2.5**
	CC-LS	0.2028 3	0.6453 2.5	**0.3868 1**	0.1866 5	2.875
	LSF-CC-PL	0.2031 5	0.6448 4	0.3649 3	0.1945 3	3.75
	LSF-CC-AL	0.2015 2	**0.6460 1**	0.3647 4	0.1936 4	2.75
Slashdot	BR	0.0165 5	0.7728 5	0.0857 4	0.6922 5	4.75
	CC	0.0168 6	0.7711 6	0.0817 6	0.6854 6	6
	CC-LO	0.0158 4	0.7838 4	0.0846 5	0.7044 4	4.25
	CC-LS	0.0157 3	0.7841 3	**0.0892 1**	0.7057 3	2.5
	LSF-CC-PL	**0.0154 1**	**0.7900 1**	0.0875 3	**0.7094 1**	**1.5**
	LSF-CC-AL	0.0156 2	0.7875 2	0.0876 2	0.7073 2	2
Genbase	BR	0.0011 5	0.9872 5	**0.7285 1**	0.9713 5	4
	CC	0.0013 6	0.9857 6	0.7202 6	0.9683 6	6
	CC-LO	0.0007 2	0.9921 2	0.7237 3	0.9819 2	2.25
	CC-LS	**0.0006 1**	**0.9927 1**	0.7244 2	**0.9834 1**	**1.25**
	LSF-CC-PL	0.0008 3.5	0.9905 3.5	0.7013 4.5	0.9758 3.5	3.75
	LSF-CC-AL	0.0008 3.5	0.9905 3.5	0.7013 4.5	0.9758 3.5	3.75

In the first group of experiments, Tables 4 and 5 report the detailed experimental results with SVM as base classifier among our proposed algorithms and those algorithms of problem transformation, i.e. CC-LS, CC-LO and BR. In order to compare their effectiveness more intuitively, we present the average rank of each algorithm in Figures 2 and 3. In Figure 2, for each metric, the average rank of each algorithm over all data sets is depicted. For example, the ranks of Hamming Loss of LSF-CC-LS on each data set are {1, 3, 1, 2, 2, 5, 1, 3.5}, then the average rank of LSF-CC-LS for Hamming Loss is $(1+3+1+2+2+5+1+3.5)/8 = 2.3125$. While in Figure 3, for each algorithm, its overall average rank over all experiments is depicted. For example, the average ranks of LSF-CC-LS on each data set are {2.25, 2, 2.75, 2.875, 3, 3.75, 1.5, 3.75}, then the overall average rank of it is $(2.25+2+2.75+2.875+3+3.75+1.5+3.75)/8 = 2.73$. According to these experimental results, we have the following remarks:

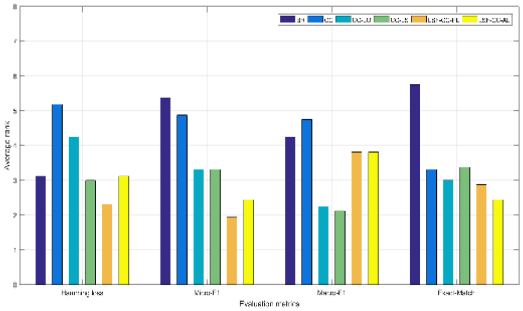


FIGURE 2. The average rank of each algorithm (with SVM as a base classifier) across all data sets in terms of all evaluation metrics. The lower the rank, the better the performance.

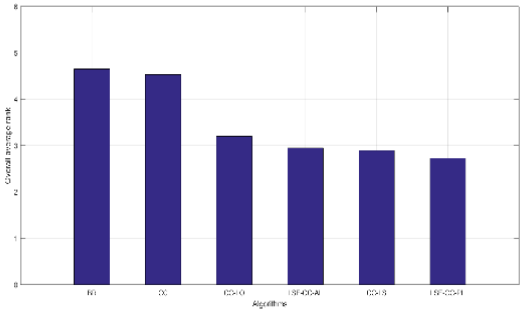


FIGURE 3. The overall average rank of all six algorithms (with SVM as a base classifier). The lower the rank, the better the performance.

1) We compare the performances between CC and its variants, i.e. CC-LO and CC-LS to reveal their shortcomings. To sum up, the performance of CC is the worst, whereas, CC-LS performs best. Specifically, according to Tables 4 and 5, CC shows the worst performances on Emotions, Language log, Enron, Yeast, Slashdot and Genbase than CC-LO and CC-LS obviously. CC-LS outperforms CC and CC-LO on Emotions, Enron, Slashdot and Genbase. CC-LO obtains the best performance on Image, Language log and Yeast. Figure 2 also shows that for all evaluation metrics, CC performs worst and CC-LS performs best. This shows that the classification performances of CC can be improved by optimizing the label ordering or removing the redundancy in the label space at the same time.
2) According to Figure 2, LSF-CC-PL and LSF-CC-AL improve the performances of CC on all evaluation metrics. These experimental results also verify that the performance of CC can be improved by optimizing the label ordering and removing the redundancy in the feature space.
3) On the three data sets Image, Emotions and Slashdot, LSF-CC-PL exceeds LSF-CC-AL significantly, while on the data set Yeast, LSF-CC-AL exceeds LSF-CC-PL significantly. For the other four data sets, their performances are equal or very close. According to Figure 3, in general, LSF-CC-PL outperforms LSF-CC-AL on the overall performances. According to these results,

we can conclude that it is important for classifier chain based algorithms to select features from not only feature space but also label space.

4) Through a comprehensive comparison among all algorithms, we can find that LSF-CC-PL performs best, followed by CC-LS and LSF-CC-AL, and then the other three are the worst. This fully proves that the proposed algorithm is very effective in improving the traditional CC algorithm via label specific features and label ordering optimization. This also shows that although CC-LO and CC-LS achieve performance improvement by adapting CC with label ordering optimization and removing the redundancy in label space, its performance can be further improved by extracting the label specific features from the feature space.

TABLE 6. Results of each comparing algorithm with 5-NN as a base classifier on Image, Emotions, Medical and Language log.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Mi-F1 ↑	Ma-F1 (↑)	Exact-Match ↑	Avg. rank
Image	BR	**0.1749 1**	0.5927 6	0.5926 6	0.4385 6	4.75
	CC	0.1863 6	0.5970 4	0.5970 4	0.4935 2	4
	CC-LO	0.1788 3	0.6137 2	0.6173 2	**0.5015 1**	**2**
	CC-LS	0.1859 5	0.5956 5	0.5957 5	0.4700 5	5
	LSF-CC-PL	0.1759 2	**0.6174 1**	**0.6189 1**	0.4825 4	**2**
	LSF-CC-AL	0.1827 4	0.6082 3	0.6133 3	0.4905 3	3.25
Emotions	BR	0.2071 2	0.6481 4	0.6295 4	0.2934 5	3.75
	CC	0.2177 4	0.6559 3	0.6422 3	0.2969 3	3.25
	CC-LO	0.2251 5	0.6359 6	0.6256 6	0.2935 4	5.25
	CC-LS	0.2254 6	0.6361 5	0.6275 5	0.2833 6	5.5
	LSF-CC-PL	**0.2027 1**	**0.6735 1**	**0.6519 1**	**0.3053 1**	**1**
	LSF-CC-AL	0.2086 3	0.6675 2	0.6494 2	0.2985 2	2.25
Medical	BR	0.0168 4.5	0.6445 5	0.1758 6	0.4755 6	5.375
	CC	0.0178 6	0.6372 6	0.1812 5	0.5030 5	5.5
	CC-LO	0.0163 3	0.6672 3	**0.2018 1**	0.5112 3	2.5
	CC-LS	0.0168 4.5	0.658 4	0.1967 2	0.5041 4	3.625
	LSF-CC-PL	0.0160 2	0.6741 2	0.1903 4	**0.5265 1**	2.25
	LSF-CC-AL	**0.0159 1**	**0.6751 1**	0.1920 3	0.5214 2	**1.75**
Language log	BR	0.1727 4	0.3755 3	0.2140 4	0.1623 4.5	3.875
	CC	0.1771 5	0.3273 6	0.1953 6	0.1521 6	5.75
	CC-LO	0.1714 3	0.3729 5	0.2122 5	0.1644 2.5	.875
	CC-LS	0.1712 2	0.3749 4	0.2157 3	0.1644 2.5	2.875
	LSF-CC-PL	0.1780 6	**0.5143 1**	**0.2845 1**	**0.1685 1**	**2.25**
	LSF-CC-AL	**0.1592 1**	0.4991 2	0.2733 2	0.1623 4.5	2.375

TABLE 7. Results of each comparing algorithm with 5-NN as a base classifier on Enron, Yeast, Slashdot and Genbase.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Mi-F1 ↑	Ma-F1 (↑)	Exact-Match ↑	Avg. rank
Enron	BR	0.0583 3	0.3780 3	0.1016 6	0.0987 6	4.5
	CC	0.0619 4	0.3575 5	0.1035 4	0.1152 3	4
	CC-LO	0.0641 5.5	0.3586 4	0.1045 3	0.1075 5	4.375
	CC-LS	0.0641 5.5	0.3547 6	0.1025 5	0.1104 4	5.125
	LSF-CC-PL	0.0567 2	**0.4791 1**	**0.1252 1**	**0.1328 1**	**1.25**
	LSF-CC-AL	**0.0564 1**	0.4749 2	0.1233 2	0.1281 2	1.75
Yeast	BR	0.2092 2	0.6329 3.5	0.4203 5	0.1986 6	4.125
	CC	0.2207 6	0.6262 6	0.4162 6	0.2304 5	5.75
	CC-LO	0.2128 4	0.6329 3.5	0.4345 2	0.2421 2	2.875
	CC-LS	0.2114 3	0.6378 2	**0.4379 1**	0.2342 4	2.5
	LSF-CC-PL	**0.2091 1**	**0.6392 1**	0.4340 3	0.2388 3	**2**
	LSF-CC-AL	0.2144 5	0.6308 5	0.4284 4	**0.2428 1**	3.75
Slashdot	BR	0.0315 4	0.4234 4	0.0246 3	0.4225 5	4
	CC	0.0329 6	0.3848 6	0.0225 6	0.3995 6	6
	CC-LO	0.0317 5	0.4117 5	0.0233 5	0.4246 4	4.75
	CC-LS	0.0310 3	0.4315 3	0.0243 4	0.4349 3	3.25
	LSF-CC-PL	**0.0179 1**	**0.7632 1**	**0.0407 1**	**0.6674 1**	**1**
	LSF-CC-AL	0.0181 2	0.7585 2	0.0403 2	0.6666 2	2
Genbase	BR	**0.0011 1**	**0.9872 1**	**0.7285 1**	**0.9713 1**	**1**
	CC	0.0013 2	0.9857 2	0.7202 2	0.9683 2	2
	CC-LO	0.0027 6	0.9704 6	0.6553 5	0.9517 5	5.5
	CC-LS	0.0024 4.5	0.9736 4	0.6586 4	0.9548 4	4.125
	LSF-CC-PL	0.0023 3	0.9752 3	0.6657 3	0.9500 6	3.75
	LSF-CC-AL	0.0024 4.5	0.9735 5	0.6414 6	0.9562 3	4.625

Tables 6 and 7 report the detailed experimental results with 5-NN as base classifier among our proposed algorithms

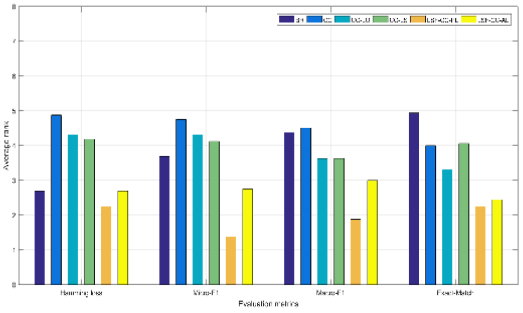


FIGURE 4. The average rank of each algorithm (with 5-NN as a base classifier) across all data sets in terms of all evaluation metrics. The lower the rank, the better the performance.

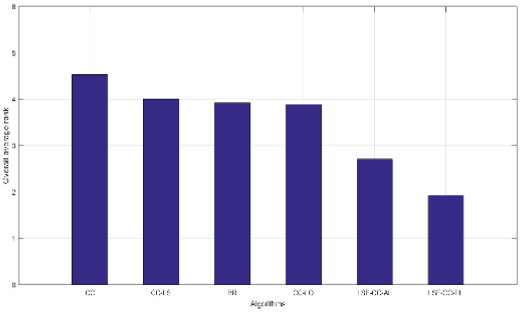


FIGURE 5. The overall average rank of all six algorithms (with 5-NN as a base classifier). The lower the rank, the better the performance.

and those algorithms of problem transformation, i.e. CC-LS, CC-LO and BR. We can easily find the superiority of our proposed algorithm. For example, on the eight data sets, LSF-CC-PL achieves better performance than all the compared methods on Image, Emotions, Language log, Enron, Yeast and Slashdot, while LSF-CC-AL achieves the best performance on Medical. On Emotions, Language log, Enron and Slashdot its performance is close to LSF-CC-PL. Furthermore, we present the average rank of each algorithm in Figures 4 and 5. From these results, we can draw similar conclusions as mentioned above, and the superiority of our proposed algorithm is more prominent. In general, for each evaluation metric, our proposed algorithm LSF-CC-PL performances best, followed by LSF-CC-AL. From the perspective of overall performance, LSF-CC-PL also is apparently higher than other algorithms, and followed by LSF-CC-AL.

In the second group of experiments, Tables 8 and 9 report experimental results among our proposed algorithms and three algorithms of algorithm transformation, i.e. MLkNN, LPLC, and LLSF. We can observe that the overall performances of LSF-CC-PL are better than that of the other compared methods. LSF-CC-PL achieves the best performances over five data sets Image, Emotions, Enron, Yeast and Slashdot. Of all the compared algorithms, the performances of LSF-CC-AL is the closest to that of algorithm LSF-CC-PL. These experimental results testify the competitive performance of our proposed algorithm comparing with the state-of-the-art algorithm transformation methods.

TABLE 8. Results of our proposed algorithms (with svm as a base classifier) and ML*k*NN, LPLC and LLSF on data sets Image, Emotions, Medical and Languagelog.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Micro-F1 ↑	Macro-F1 (↑)	Exact-Match ↑	Avg. rank
Image	ML*k*NN	**0.1714 1**	0.5907 2	0.5897 2	0.4250 3	2
	LPLC	0.2517 5	0.5533 4	0.5569 4	0.2910 4	4.25
	LLSF	0.2183 4	0.2927 5	0.2861 5	0.1505 5	4.75
	LSF-CC-PL	0.1747 2	**0.5988 1**	**0.5960 1**	0.4415 2	**1.5**
	LSF-CC-AL	0.1917 3	0.5871 3	0.5867 3	**0.4570 1**	2.5
Emotions	ML*k*NN	**0.1942 1**	0.6641 2	0.6300 3	0.2934 3	2.25
	LPLC	0.3474 5	0.6151 4	0.6135 4	0.0860 5	4.5
	LLSF	0.2561 4	0.3784 5	0.3636 5	0.1129 4	4.5
	LSF-CC-PL	0.2030 2	**0.6693 1**	**0.6582 1**	0.3018 2	**1.5**
	LSF-CC-AL	0.2085 3	0.6612 3	0.6481 2	**0.3239 1**	2.25
Medical	ML*k*NN	0.0155 3	0.6747 3	0.2107 4	0.5051 4	3.5
	LPLC	0.0194 4	0.6321 5	0.1858 5	0.5144 3	4.25
	LLSF	0.0207 5	0.6737 4	0.3545 2	0.4724 5	4
	LSF-CC-PL	**0.0097 1**	**0.8116 1**	0.3521 3	0.6810 2	1.75
	LSF-CC-AL	0.0098 2	0.8105 2	**0.3551 1**	**0.6830 1**	**1.5**
Languagelog	ML*k*NN	**0.1567 1**	**0.5383 1**	0.2677 3	0.1568 4	**2.25**
	LPLC	0.1856 4	0.4864 3	**0.3975 1**	0.1623 3	2.75
	LLSF	0.2606 5	0.4114 5	0.3292 2	0.1527 5	4.25
	LSF-CC-PL	0.1627 3	0.4847 4	0.2053 5	**0.1712 1.5**	3.375
	LSF-CC-AL	0.1618 2	0.4871 2	0.2089 4	**0.1712 1.5**	2.375

TABLE 9. Results of our proposed algorithms (with svm as a base classifier) and ML*k*NN, LPLC and LLSF on data sets Enron, Yeast, Slashdot and Genbase.

Data sets	Algorithms	Measures				
		Hamming Loss ↓	Micro-F1 ↑	Macro-F1 (↑)	Exact-Match ↑	Avg. rank
Enron	ML*k*NN	0.0523 3	0.4683 3	0.0842 5	0.0634 5	4
	LPLC	0.0790 4	0.3889 4	**0.1325 1**	0.0699 3	3
	LLSF	0.1514 5	0.2575 5	0.1241 2	0.0652 4	4
	LSF-CC-PL	**0.0503 1**	**0.5289 1**	0.1076 4	0.1128 2	**2**
	LSF-CC-AL	0.0507 2	0.5256 2	0.1077 3	**0.1146 1**	**2**
Yeast	ML*k*NN	**0.1974 1**	0.6397 3	0.3632 4	0.1808 4	3
	LPLC	0.4276 5	0.5659 4	**0.4577 1**	0.0198 5	3.75
	LLSF	0.2412 4	0.3932 5	0.1899 5	0.0405 3	4.25
	LSF-CC-PL	0.2031 3	0.6448 2	0.3649 2	**0.1945 1**	**2**
	LSF-CC-AL	0.2015 2	**0.6460 1**	0.3647 3	0.1936 2	**2**
Slashdot	ML*k*NN	0.0224 4	0.6978 3	0.0368 4	0.5973 4	3.75
	LPLC	0.0370 5	0.3169 5	0.0228 5	0.3451 5	5
	LLSF	0.0208 3	0.6748 4	0.0847 3	0.6111 3	3.25
	LSF-CC-PL	**0.0154 1**	**0.7900 1**	0.0875 2	**0.7094 1**	**1.25**
	LSF-CC-AL	0.0156 2	0.7875 2	**0.0876 1**	0.7073 2	1.75
Genbase	ML*k*NN	0.0048 5	0.9456 3	0.5340 5	0.9110 5	4.5
	LPLC	0.0031 4	0.9650 2	0.6133 4	0.9350 4	3.5
	LLSF	**0.0008 2**	**0.9910 1**	**0.7549 1**	**0.9773 1**	**1.25**
	LSF-CC-PL	**0.0008 2**	0.9905 4.5	0.7013 2.5	0.9758 2.5	2.875
	LSF-CC-AL	**0.0008 2**	0.9905 4.5	0.7013 2.5	0.9758 2.5	2.875

V. CONCLUSION

The traditional classifier chain based algorithm CC randomly selects label order and then learns the corresponding binary classifier one by one. For the current label, the learning algorithm takes the original feature space and all the labels before it in the chain as input, and utilizes the higher-order label relationship by this way. we discusses two disadvantages of CC that affect the classification performance: random label order, original feature and new feature noises. In order to overcome these shortcomings, a novel and effective classifier chain based algorithm LSF-CC is proposed. There are two key ideas in it. First, an optimized label order is determined by using the label correlations. The principle is that if a label is more related with other labels, it should be learned more early. After the label dependencies are calculated using the ReliefF method, the label order is determined according to this principle. Secondly, for the current label, the label specific features are selected from the original feature space and the label space respectively, and the corresponding binary classifier is learned according to these label specific features. Experimental results show that our proposed algorithm is effective. LSF-CC not only solves the shortcomings of traditional classifier chain based algorithm, but also provides a general framework for further improving the CC model. Within this framework, it is possible to further explore how to use other theoretical and technical means to determine label correlations and label specific features.

REFERENCES

[1] T.-Y. Wang and H.-M. Chiang, "Solving multi-label text categorization problem using support vector machine approach with membership function," *Neurocomputing*, vol. 74, no. 17, pp. 3682–3689, Oct. 2011.

[2] J. Y. Lin, Q. Su, P. C. Yang, S. M. Ma, and X. Sun, "Semantic-unit-based dilated convolution for multi-label text classification," in *Proc. 2018 Conf. Empirical Methods Natural Lang. Process.*, Aug. 2018, pp. 4554–4564.

[3] Y. Luo, T. Liu, D. Tao, and C. Xu, "Multiview matrix completion for multilabel image classification," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2355–2368, Aug. 2015.

[4] S. Shi, Y. Chen, M. Fang, W. Li, and Shining, "A hierarchical multi-label propagation algorithm for overlapping community discovery in social networks," in *Proc. 11th Web Inf. Syst. Appl. Conf.*, Sep. 2014, pp. 113–118. [Online]. Available: https://dblp.uni-trier.de/pers/hd/s/Shi:Song

[5] Y. Hua, L. Mou, and X. X. Zhu, "Recurrently exploring class-wise attention in a hybrid convolutional and bidirectional LSTM network for multi-label aerial image classification," *ISPRS J. Photogram. Remote Sens.*, vol. 149, pp. 188–199, Mar. 2019.

[6] 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, Aug. 2014.

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

[8] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proc. ICML*, 2010, pp. 1609–1614.

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

[10] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "Correlation-based pruning of stacked binary relevance models for multi-label learning," in *Proc. 1st Int. Workshop Learn. Multi-Label Data*, Sep. 2009, pp. 101–116.

[11] W. Weng, C.-L. Chen, S.-X. Wu, Y.-W. Li, and J. Wen, "An efficient stacking model of multi-label classification based on Pareto optimum," *IEEE Access*, vol. 7, pp. 127427–127437, 2019.

[12] A. Yuce, H. Gao, and J.-P. Thiran, "Discriminant multi-label manifold embedding for facial action unit detection," in *Proc. 11th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, May 2015, pp. 1–6.

[13] Y. Zhu, J. T. Kwok, and Z.-H. Zhou, "Multi-label learning with global and local label correlation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 6, pp. 1081–1094, Jun. 2018.

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

[15] Y. Lin, Q. Hu, J. Liu, and J. Duan, "Multi-label feature selection based on max-dependency and min-redundancy," *Neurocomputing*, vol. 168, pp. 92–103, Nov. 2015.

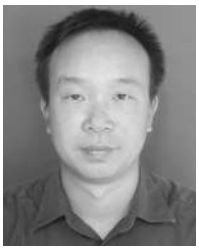
[16] P. Naula, A. Airola, T. Salakoski, and T. Pahikkala, "Multi-label learning under feature extraction budgets," *Pattern Recognit. Lett.*, vol. 40, pp. 56–65, Apr. 2014.

[17] J. Huang, G. Li, Q. Huang, and X. Wu, "Joint feature selection and classification for multilabel learning," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 876–889, Mar. 2018.

[18] 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, Jan. 2015.

[19] J. Huang, X. Qu, G. Li, F. Qin, X. Zheng, and Q. Huang, "Multi-view multi-label learning with View-Label-Specific features," *IEEE Access*, vol. 7, pp. 100979–100992, 2019.
[20] J. Huang, F. Qin, X. Zheng, Z. Cheng, Z. Yuan, W. Zhang, and Q. Huang, "Improving multi-label classification with missing labels by learning label-specific features," *Inf. Sci.*, vol. 492, pp. 124–146, Aug. 2019.
[21] W. Weng, Y.-N. Chen, C.-L. Chen, S.-X. Wu, and J.-H. Liu, "Non-sparse label specific features selection for multi-label classification," *Neurocompting*. vol. 377, pp. 85–94, Feb. 2020, doi: 10.1016/j.neucom.2019.10.016.
[22] Y. Yan, S. Li, Y. Zhe, Z. Xiao, L. Jing, A. Wang, and J. Zhang, "Multi-label learning with label specific feature selection," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 305–315.
[23] J. Huang, G. Li, Q. Huang, and X. Wu, "Learning label specific features for multi-label classification," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 181–190.
[24] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 135–168, 2000.
[25] S. Godbole and S. Sarawagi, "Discriminative methods for multilabeled classification," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springe, 2004, pp. 22–30.
[26] E. Alvares-Cherman, J. Metz, and M. C. Monard, "Incorporating label dependency into the binary relevance framework for multi-label classification," *Expert Syst. Appl.*, vol. 39, no. 2, pp. 1647–1655, Feb. 2012.
[27] E. C. Goncalves, A. Plastino, and A. A. Freitas, "Simpler is better: A novel genetic algorithm to induce compact multi-label chain classifiers," in *Proc. 2015 Annu. Conf. Genetic Evol. Comput.*, Jul. 2015, pp. 559–566.
[28] G. Tsoumakas and I. Vlahavas, "Random *k*-labelsets: An ensemble method for multilabel classification," in *Proc. 18th Eur. Conf. Mach. Learn. (ECML)*, 2007, pp. 406–417.
[29] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 995–1000.
[30] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," *Artif. Intell.*, vol. 172, nos. 16–17, pp. 1897–1916, Nov. 2008.
[31] E. Hüllermeier, J. Fürnkranz, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Mach. Learn.*, vol. 73, no. 2, pp. 133–153, Nov. 2008.
[32] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Mach. Learn.*, vol. 73, no. 2, pp. 185–214, Nov. 2008.
[33] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 389–396, 2011.
[34] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
[35] 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, Oct. 2006.
[36] A. Elisseeff and J. Westom, "A kernel method for multi-labelled classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, USA: MIT Press, vol. 14, 2001, pp. 681–687.
[37] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Proc. 5th Eur. Conf. Princ. Data Mining Knowl. Discovery*, Freiburg, Germany, 2001, pp. 42–53.
[38] I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1994, pp. 171–182.
[39] H. Y. Liu, Z. H. Wang, and Y. Sun, "Stacking model of multi-label classification based on pruning strategies," *Neural Comput. Appl.*, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s00521-018-3888-0#citeas, doi: 10.1007/s00521-018-3888-0.
[40] N. Spolaôr, E. A. Cherman, M. C. Monard, and H. D. Lee, "Filter approach feature selection methods to support multi-label learning based on ReliefF and information gain," in *Proc. 21st Brazilian Conf. Adv. Artif. Intell.*, 2012, pp. 72–81.
[41] E. L. Gibaja, S. Ventura, "A tutorial on multi-label learning," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 1–38, 2015.
[42] J. Huang, G. Li, S. Wang, and Q. Huang, "Categorizing social multimedia by neighborhood decision using local pairwise label correlation," in *Proc. IEEE Int. Conf. Data Mining Workshop*, Dec. 2014, pp. 913–920.

WEI WENG received the M.S. degree from the Department of Computer Science and Engineering, Xiamen University, Xiamen, China, in 2005, and the Ph.D. degree from the School of Aerospace Engineering, Xiamen University, in 2019. He is currently an Associate Professor with the School of Computer and Information Engineering, Xiamen University of Technology. His research interests include network security, mobile computing, and electronic commerce.



DA-HAN WANG received the B.S. degree in automation science and electrical engineering from Beihang University, Beijing, China, in 2006, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2012. He is currently an Associate Professor with the School of Computer and Information Engineering, Xiamen University of Technology, and the Fujian Key Laboratory of Pattern Recognition and Image Understanding. His research interests include pattern recognition, text detection and recognition, fine-grained image classification, and semantic segmentation.

CHIN-LING CHEN received the Ph.D. degree from National Chung Hsing University, Taiwan, in 2005. From 1979 to 2005, he was a Senior Engineer with Chunghwa Telecom Company, Ltd. He is a distinguished professor. He has published over 90 articles in SCI/SSCI international journals. His research interests include cryptography, network security, and electronic commerce.

JUAN WEN received the M.S. degree from the Department of Computer Science, Xiamen University, in 2006, and the Ph.D. degree from the Department of Statistics, Xiamen University, in 2012. She is currently an Assistant Professor with the Department of Statistics, School of Economics, Xiamen University. Her research interests include statistical analysis and data mining.

SHUN-XIANG WU was born in Shaoyang, China, in 1967. He received the M.S. degree from the Department of Computer Science and Engineering, Xi'an Jiaotong University, in 1991, and the Ph.D. degree from the School of Economics and Management, Nanjing University of Aeronautics and Astronautics, in 2007. He is currently a Professor with the Department of Automation, School of Aerospace Engineering, Xiamen University. His research interests include intelligent computing, data mining and knowledge discovery, and systems engineering theory and its application.