Information Sciences 351 (2016) 101–114

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An approach for multi-label classification by directed acyclic graph with label correlation maximization

Jaedong Lee[a], Heera Kim[b], Noo-ri Kim[a], Jee-Hyong Lee[a,*]

[a] Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea
[b] Department of Platform Software, Sungkyunkwan University, Suwon 16419, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 7 August 2015
Revised 12 January 2016
Accepted 18 February 2016
Available online 2 March 2016

Keywords:
Multi-label learning
Directed acyclic graph
Bayesian network
Conditional entropy

A B S T R A C T

Traditional supervised learning approaches primarily work in the single-label environment. However, in many real-world problems, data instances are usually associated with multiple labels simultaneously, and multi-label learning is increasingly required in many modern applications. In multi-label learning, the key to successful classification is effectively exploiting the complex correlations among the output labels. This paper proposes a novel multi-label learning method inspired by the classifier chain approach. The main contribution of this work is to model the correlations of the labels using a directed acyclic graph. Starting from the simple intuitive notion of measuring the correlations among the labels, the proposed method is designed as a multi-label learning method that maximizes the correlations among labels. To evaluate its effectiveness, the proposed method is compared with the state-of-the-art approaches. Extensive experiments demonstrated the proposed method to be highly competitive with the other multi-label approaches.

1. Introduction

Traditional supervised learning approaches primarily work in the single-label environment, where an instance is associated with only a single label. However, in many real-world conditions, data instances are usually associated with multiple labels simultaneously. Thus, multi-label learning is being increasingly required in many modern applications. For example, an image in an image-tagging system could have conceptual labels of both “tree” and “forest” [3,17,26]. Similarly, in text categorization, a document can belong to multiple categories or have multiple tags [1,12,24].

The goal of single-label learning is to find a model $\boldsymbol{h}$ that maps input $\mathbf{x}$ to a scalar output y. In contrast, in multi-label learning, the problem is to find a model $\boldsymbol{h}$ that maps input $\mathbf{x}$ to a vector output $\mathbf{y} = (y_1, y_2, \ldots, y_L)$. If the values of y_i are uncorrelated, that is, if knowing the values of y_j for $j \neq i$ is not helpful for prediction of y_i, then multi-label learning is equivalent to L single-label learnings in a straightforward way. In this case, the classifier for each label can be independently built. However, output labels are correlated with each other in most cases, which means that y_i can be better predicted if some values of y_j for $j \neq i$ are known. This introduces complexity that makes the task of multi-label learning rather challenging. This viewpoint has motivated much multi-label learning research, which has led to the development of various methods to utilize correlation information to improve the performance of classifiers [2,4,6,7,10,15,18,20,22,23,25,28–32].

The existing multi-label learning approaches can be grouped into two categories: label power set (LP) approaches and binary relevance (BR) approaches. In LP approaches [16,20,21,27,28,30], a multi-label problem is transformed into a

* Corresponding author. Tel.: +82 31 290 7154.
E-mail addresses: ultrajaepo@skku.edu (J. Lee), hrahoy@skku.edu (H. Kim), pd99j@skku.edu (N.-r. Kim), john@skku.edu (J.-H. Lee).

http://dx.doi.org/10.1016/j.ins.2016.02.037
0020-0255/© 2016 Elsevier Inc. All rights reserved.

Table 1
Example of multi-label data.

Inputs	Outputs
$\mathbf{x}_1 = [1, 0, 1, 1, 0, 0, 1]$	$\mathbf{y}_1 = [1, 0, 0, 1, 0]$
$\mathbf{x}_2 = [0, 0, 1, 1, 0, 1, 0]$	$\mathbf{y}_2 = [0, 1, 0, 1, 1]$
$\mathbf{x}_3 = [1, 0, 0, 1, 1, 1, 1]$	$\mathbf{y}_3 = [1, 0, 1, 0, 1]$

multi-class single-label problem by treating an output vector as a scalar value. Then, the problem can be directly solved using any single-label learning approach. LP approaches have the advantage that the correlations of labels can be embedded in scalar values and learned by single-label learners, but they also have some drawbacks: high computational complexity and overfitting to the training data [28]. To overcome these disadvantages, several methods have been proposed based on randomized approaches [16,20,21,27,28,30].

Additionally, BR is another common approach, which transforms a multi-label problem into multiple independent single-label classification problems. In BR, all labels are assumed to be independent of themselves, and each classifier is independently trained to predict each label based on only given inputs. However, this assumption of label independence is not suitable for real-world multi-label dataset. So, in order to consider the correlations of labels, there were some variations of BR that were proposed.

The classifier chain (CC) approach based on BR was introduced by Read et al. [22]. They proposed the idea of “chaining” classifiers to take the label correlations into account. This method involved L single-label classifiers linked along a chain in a specific order; for example, $\{y_2 \rightarrow y_5 \rightarrow \ldots \rightarrow y_1\}$. The inputs of each classifier in a chain were extended with the outputs of all the preceding classifiers. CC approaches can properly model correlations among labels and have been shown to be competitive in terms of their computational efficiency [8]. However, they have a critical drawback in that it is very hard to determine the proper order of classifier chains, which led to the decision to employ a random-order scheme [7,11,14,23,25]. Many extensions of CC approaches have been proposed to deal with the ordering problem. Read et al. proposed ensemble classifier chains (ECC) in an effort to reduce the influence of bad random chain order by adopting a simple ensemble framework [23]; Dembczynski et al. introduced a probabilistic method to estimate the entire joint distribution of the label set in the classification phase [7]. Silva et al. built several randomly ordered chains and then tried to find the best one for a test instance using a brute-force method [25]. Gonçalves et al. adopted a genetic algorithm on CC for label ordering optimization [11]. However, CC approaches still have limitations. Previous work has focused on avoiding bad ordered chains rather than finding the optimal order to improve prediction performance.

Successful multi-label learning requires effectively exploiting the complex correlations among the output labels. We thus propose a novel multi-label learning method inspired by classifier chain approaches. Our proposed method overcomes some limitations of the existing multi-label classification approaches by finding the order of the classifiers that maximizes the correlations between the labels. The main contribution of this work is to build a directed acyclic graph (DAG) of labels in which the correlations between parent and child nodes are maximized. We quantify the correlations with the conditional entropy and find a DAG that maximizes the sum of conditional entropies between all parent and child nodes. Thus, highly correlated labels can be sequentially ordered in chains obtained from the DAG, and the prediction results can be optimized when utilizing a CC approach with the chains. What is interesting is that finding such a DAG is very similar to the Bayesian network learning problem. Thus, we can solve this problem by using one of several Bayesian network learning methods, such as the K2 algorithm.

The rest of this paper is organized as follows. Section 2 introduces the multi-label classification approach and discusses related work. Section 3 presents a detailed description of the proposed multi-label learning algorithm. Section 4 compares the proposed multi-label learning method to well-known multi-label classification algorithms. The multi-label dataset and evaluation measures used for experiments are also described in detail. Finally, Section 5 highlights the main points of this work and concludes this paper.

2. Related work

In this section, label power set (LP) approaches and binary relevance (BR) approaches for the multi-label learning problem are summarized.

2.1. Label power set approach

An instance in multi-label data is composed of a vector of inputs $\mathbf{x}$ and a vector of outputs $\mathbf{y}$. Table 1 lists 3 instances of multi-label data. There are 7 inputs and 5 outputs: $\mathbf{x} = (x_1, x_2, \ldots, x_7)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_5)$. The goal of multi-label learning is to find a model that maps an input vector $\mathbf{x}$ to a vector output $\mathbf{y}$.

In the LP approach, a multi-label problem is transformed into a multi-class single-label problem by treating an output vector as a scalar value. Then, the problem is to find a model $\boldsymbol{h}$ that maps an input vector $\mathbf{x}$ to a scalar output z representing an output vector $\mathbf{y}$. For example, $\mathbf{y}_1$ in Table 1 can be transformed into a binary number $z_1 = 10010$, and similarly, $\mathbf{y}_2$ into

$z_2 = 01011$ and $\mathbf{y}_3$ into $z_3 = 10101$. In this case, a classifier $\boldsymbol{h}$ aims to predict a scalar output z from 7 dimensional inputs. This makes it possible to directly apply single-label classifiers.

The LP approach has the advantages that it creates only one classification model $\boldsymbol{h}$ and that it can directly learn the correlations between labels from the dataset. However, the LP approach has several disadvantages. First, an obvious drawback is that it can have a high computational complexity problem. Because the output vectors in the dataset are modeled with different scalar values, the number of possible scalar values of z increases too much. Another crucial limitation is that the LP method can only predict label values that have been previously observed in the training dataset. For example, since a label value $z' = 10101$ does not appear in Table 1, it cannot be a candidate output of a classification model trained with the data in the table, which means no inputs can be classified as z'. In this case, the model tends to overfit the training data [28].

To overcome this disadvantage, several LP-based approaches have been proposed. Read et al. proposed the pruned sets (PS) method and the Ensemble of Pruned Sets (EPS) method [20]. By pruning away infrequently occurring label sets, the PS method tried to train the correlations only among a major label set while reducing the computational complexity. EPS combined the results of several pruned sets in an ensemble scheme to predict unseen labels. Read et al. also proposed a general framework for LP approaches with meta-labels in which a variety methods for pruning, partitioning and ensemble learning could be easily evaluated and combined into novel methods [21].

Tsoumakas et al. proposed a randomized algorithm called RAkEL (random k-label subsets), utilizing the co-occurrence of labels in the dataset [30]. RAkEL randomly split the initial L labels into m subsets of size k. For example, 5 labels could be split into $\{y_1, y_3, y_4\}$, $\{y_2, y_3, y_5\}$, $\{y_1, y_4, y_5\}$ with duplications if $m = 3$ and $k = 3$. Each subset was modeled with an LP model, and the final decision was made using a simple ensemble-based voting process. They also proposed a hierarchical multi-label classifier named HOMER [29]. HOMER first clustered the labels into k balanced disjoint subsets, and each subset of labels was trained by an LP approach. For example, if $k = 2$, then 5 labels could be clustered into 2 subsets, such as $\{y_2, y_3, y_5\}$ and $\{y_1, y_4\}$, with a clustering algorithm and each subset was trained by LP models. Finally, all labels in each subset were trained by single-label classifiers.

Lo et al. proposed a generalized k-label sets ensemble called GLE [16]. GLE used random k-label set LP classifiers as base learners. The objective function was designed to learn the coefficients of the base classifiers by minimizing the global error with a hypergraph Laplacian regularization term. They also extended the GLE for cost-sensitive multi-label classification in order to minimize the misclassification costs for all labels jointly.

Tahir et al. proposed an instance-based approach with a dimensionality reduction for multi-label classification [27]. They used the stacked kernel discriminant analysis for the dimensionality reduction and multi label k nearest neighbor (MLkNN) approach for the classification. They extracted a small number of features that considered label correlations among different labels. Using new features, they chose nearest neighbors for an instance and determined its labels considering the label probability distributions in the nearest neighbors.

Inherently, the LP methods can directly learn the label correlations from the data and use them to improve the predictions. However, most of them have some disadvantages, such as high computation complexity and overfitting. Some researchers have proposed improvements, but their approaches were based on randomness or statistical techniques, rather than trying to directly capture the label correlations by analyzing the data.

2.2. Binary relevance approach

Another common approach for multi-label learning is the binary relevance approach (BR) [4]. In the BR approach, a multi-label problem is transformed into multiple single-label problems. In other words, all labels are assumed to be mutually independent. Thus, classifiers for each label are independently trained based only on the given inputs. Since simple BR approaches are incapable of considering the correlations between labels, some researchers have proposed extended BR approaches incorporating label correlation information [4,6,7,8,10,11,14,22,23,25,32]. They can be grouped into two categories: the meta-training (MT) approach and the classifier chain (CC) approach.

Godbole and Sarawagi first introduced the MT approach [10]. First, they built interim classifiers $\hat{h}_i$ for labels y_i without considering correlations among the labels, using a simple BR approach. This step is called the meta-training step. The final classifiers h_i are trained using the inputs together with the outputs predicted by the interim classifiers $\hat{h}_i$ as additional inputs. A simple BR approach is also adopted to build the final classifiers.

Table 2 shows an example of the training phases in the MT approach. There is one set of training data, $\mathbf{x} = [1, 0, 1, 1, 0, 0, 1]$ and $\mathbf{y} = [1, 0, 0, 1, 0]$. The five interim classifiers, $\hat{\boldsymbol{h}} = (\hat{h}_1, \hat{h}_2, \hat{h}_3, \hat{h}_4, \hat{h}_5)$, are built using the input $\mathbf{x} = [1, 0, 1, 1, 0, 0, 1]$ to predict $\mathbf{y} = [1, 0, 0, 1, 0]$. Using the predicted values of the interim classifiers, the inputs for the final classifiers are extended. If the output of the 5 interim classifiers for $\mathbf{x}$ is [1, 0, 1, 1, 0], i.e., $\hat{\boldsymbol{h}}(\mathbf{x}) = [1, 0, 1, 1, 0]$, then the input $\mathbf{x}$ is extended to $\mathbf{x}' = [\mathbf{x}; \hat{\boldsymbol{h}}(\mathbf{x})] = [1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$ by combining $\mathbf{x}$ and $\hat{\boldsymbol{h}}(\mathbf{x})$. The final classifiers, $\boldsymbol{h} = [h_1, h_2, h_3, h_4, h_5]$, are built with $\mathbf{x}'$.

A number of improved MT approaches have been proposed. Cheng and Hüllermeier proposed an instance-based learning approach named IBLR [6]. Instead of building interim classifiers, they adopted an instance-based approach using logistic regression to obtain the interim outputs for an instance. Zhang and Zhang proposed an approach exploiting label dependencies [32], which instead of using all the outputs of the interim classifiers, chose the interim classifiers $\hat{h}_\mathrm{i}$ that were strongly correlated with y_j and used them as additional inputs for the final classifier for y_j. They used the training errors of the interim classifiers to determine additional inputs of the final classifiers. Cherman et al. introduced a new method that

Table 2
Example of training phases in the MT approach for $\mathbf{x} = [1,0,1,1,0,0,1]$ and $\mathbf{y} = [1,0,0,1,0]$.

$\hat{\boldsymbol{h}}_i$	$\mathbf{x}$	y_i	$\hat{h}_i(\mathbf{x})$	$\boldsymbol{h}_i$	$\mathbf{x}' = [\mathbf{x}; \hat{\boldsymbol{h}}(\mathbf{x})]$	y_i
$\hat{h}_1$:	$[1, 0, 1, 1, 0, 0, 1]$	1	1	h_1 :	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$	1
$\hat{h}_2$:	$[1, 0, 1, 1, 0, 0, 1]$	0	0	h_2 :	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$	0
$\hat{h}_3$:	$[1, 0, 1, 1, 0, 0, 1]$	0	1	h_3 :	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$	0
$\hat{h}_4$:	$[1, 0, 1, 1, 0, 0, 1]$	1	1	h_4 :	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$	1
$\hat{h}_5$:	$[1, 0, 1, 1, 0, 0, 1]$	0	0	h_5 :	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1, 0]$	0
(a) Training data for $\hat{h}_i$ in interim phase				(b) Training data for h_i in final phase		

Table 3
Example of training phase by CC model.

$\boldsymbol{h}_i$	$\mathbf{x}'_i = [\mathbf{x}; \boldsymbol{h}_j(\mathbf{x})]$ where $j < i$	y_i	$h_i(\mathbf{x}'_i)$
h_1:	$[1, 0, 1, 1, 0, 0, 1]$	1	1
h_2:	$[1, 0, 1, 1, 0, 0, 1; 1]$	0	0
h_3:	$[1, 0, 1, 1, 0, 0, 1; 1, 0]$	0	1
h_4:	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1]$	1	1
h_5:	$[1, 0, 1, 1, 0, 0, 1; 1, 0, 1, 1]$	0	0

improved the final classification scheme [8]. After constructing the interim classifiers, this method builds a linear order of the labels based on the confidences of the interim classifiers. The final classifiers are also built in that order. For example, if the order is $\{y_2 \to y_4 \to y_3 \to y_5 \to y_1\}$, then the final classifier for y_3, h_3, is built after building h_2 and h_4 because y_3 is third. The inputs for h_i are extended with the results of the final classifiers of the preceding labels and the interim classifiers of the following labels. In this example, inputs for the final classifier h_3 are extended to $[\mathbf{x}; \hat{h}_1(\mathbf{x}), h_2(\mathbf{x}), h_4(\mathbf{x}), \hat{h}_5(\mathbf{x})]$.

MT approaches consider label correlations; however, they have some disadvantages. MT approaches entail more complex computations in the meta-training phase, so they require extra interim iterations in both the training and the test phases. Moreover, their consideration of the label correlations is ineffective because they merely use the values of interim classifiers that were built independently.

Another approach based on BR is the classifier chain (CC) method introduced by Read et al. [22]. Instead of using the outputs of the interim classifiers, CC approaches use the outputs of the final classifiers. Final classifiers are built one by one in a given order. The input for a classifier is extended with the output of its preceding classifiers. Table 3 shows a simple example using CC where input $\mathbf{x} = [1, 0, 1, 1, 0, 0, 1]$ and output $\mathbf{y} = [1, 0, 0, 1, 0]$. The order of labels is $\{y_1 \to y_2 \to y_3 \to y_4 \to y_5\}$. So h_1, a classifier for y_1, is built first using only the given inputs, and h_2 is built next using the given inputs and the output of the preceding classifier, $h_1(\mathbf{x})$. In Table 3, $h_1(\mathbf{x}) = 1$, so the input h_2 is extended to $\mathbf{x}'_2 = [1, 0, 1, 1, 0, 0, 1; 1]$. In this manner, the inputs for each classifier h_i are extended with the outputs of all the precedent classifiers. Since the predicted outputs of precedent classifiers are directly used for the following classifiers without interim classifiers, the predictive performance can be improved while maintaining a low computational complexity if a valid ordering of the classifiers chain is given. However, the authors did not propose how to find such orderings and just used a random order of labels.

Since finding effective ordering is not easy, Read et al. combined an ensemble approach with their classifier chain method (ECC) [23]. Instead of finding a good chain, they repeated their CC procedure several times with randomly generated chains of labels and determined final classifications based on all of these results.

Dembczynski introduced a probabilistic classifier chain (PCC) approach [7]. As in the simple CC approach, their method randomly determined the order of the classifier chain in the training phase. According to this order, for each instance of classification phase, it estimated the entire joint distribution of all possible label combinations and sought to maximize the posterior probability of the predicted label combination. They also proposed an ensemble PCC scheme.

Silva et al. proposed the idea of using a set of classifier chains [25]. This method began by making several classifier chains with random orders of labels. For a particular instance, this method found k-nearest neighbors of the instance in the training dataset, and sought for the classifier chain showing the best performance for the neighbors. Then, the best classifier was applied to the instance. This method aimed to find a good chain order by using brute-force.

Gonçalves et al. proposed a novel method for multi-label classifier chains based on a genetic algorithm for a single optimized label ordering [11]. This method used a variable-length list representation and a multi-objective lexicographic fitness function to evaluate chain sequences and considered both the accuracy and size of the model.

Lee and Kim tried to find a more effective feature subset by maximizing the dependency between features and labels [14]. Their method hybridized an effective local refinement method with a genetic algorithm to find a more effective feature subset for each label of dataset and providing the multi-label classifier with more discriminating capability.

Many variations of the CC approach greatly reduce the risk of poor ordering and show good predictive performances. However, most of them have the disadvantage of being based on randomness and lack analysis for a good chain order. Even though some researchers have tried to use mathematical models, their methods create high computational complexity unsuitable for large-scale multi-label datasets [25].

3. Proposed method

In multi-label classification, it is important to maximize the performance by capturing and reflecting the correlations between labels. Most other previous works have tried to solve this problem by adopting randomized approaches rather than by analyzing the data to capture the correlations. In this section, we propose a novel multi-label learning approach inspired by the classifier chain approach that can effectively model the label correlations using a directed acyclic graph by analyzing a given dataset. Additionally, it improves prediction performances by building classifiers based on the correlation model.

In CC approaches, the problem of predicting L output labels is separated into L problems each predicting a label. After the labels are linearly ordered, the outputs of the precedent classifiers are used to build the next classifiers. To make more accurate predictions, it is important to find a properly ordered chain. Whereas most previous CC approaches have sought linear orders of labels, we generalize the CC approach as solving the problem of finding a direct acyclic graph, G, of the labels. As in the CC approach, we use the predictions of the precedent labels for the construction of the following labels in G. The input of a classifier is extended with the outputs of its precedent classifiers.

Once a directed acyclic graph of the labels is built, a classifier for a label is built with the given inputs and the outputs of the classifiers of its precedent labels. To obtain high classification performance, the direct precedent labels of a label must be sufficiently correlated with the label. This is a trivial condition because the task is to find additional inputs that can help in predicting a label. If a label y_j gives useful information for predicting a label y_i, then y_i and y_j must be correlated with each other. So to construct the optimal DAG is to find the DAG that maximizes the correlation between directly connected labels. We formulate this problem as follows:

$$\underset{\mathbf{G}}{\operatorname{argmax}} \sum_{i=1}^{L} \text{Correlation of } w_i \text{ with } y_i \tag{1}$$

where there are L labels $\mathbf{y} = (y_1, y_2, \ldots, y_L)$, $\boldsymbol{G}$ is a DAG of $\mathbf{y}$, and $\mathbf{w}_i$ is the set of direct precedent labels of y_i in $\boldsymbol{G}$. To measure how much two labels are correlated, the conditional entropy is adopted, which is defined as follows:

$$H(Y|X) \equiv \sum_{x\in X} p(x)H(Y|X=x) = \sum_{x\in X,\quad y\in Y} p(x,y)\log\frac{p(x)}{p(x,y)} \tag{2}$$

If the conditional entropy of Y given X, $H(Y|X)$, is zero, the value of Y is perfectly predicted by observing the value of X. Intuitively, if $H(Y|X)$ is low enough, it means that X has retained much of the information about the value of Y, and thus, it can be said that Y is correlated with X. The best $\mathbf{w}_i$ for y_i is the one such that $H(y_i|\ \mathbf{w}_i)$ is minimized. So the final objective function can be stated as Eq. (3).

$$\begin{aligned}&\underset{\boldsymbol{G}}{\operatorname{argmax}} \textstyle\sum_{i=1}^{L} -H(y_i|\boldsymbol{w}_i)\\ &\text{where}\\ &\boldsymbol{G} \text{ is a DAG whose nodes are } \boldsymbol{y}\\ &\boldsymbol{w}_i \text{ is the set of the direct predecessors of } y_i\end{aligned} \tag{3}$$

It is worth mentioning here that finding a DAG that maximizes the sum of $-H(y_i|\boldsymbol{w}_i)$ is equivalent to learning a Bayesian network that maximizes the MLC score, as follows:

$$\begin{aligned}-\sum_{i=1}^{L} H(y_i|\boldsymbol{w}_i) &= \sum_{i=1}^{L}\sum_{j=1}^{q_i} -p(y_i)H\big(Y|\boldsymbol{w}_i=\boldsymbol{w}_{ij}\big)\\ &= \sum_{i=1}^{L}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} p\big(y_{ik},\boldsymbol{w}_{ij}\big)\log\frac{p\big(y_{ik},\boldsymbol{w}_{ij}\big)}{p\big(\boldsymbol{w}_{ij}\big)}\\ &= \sum_{i=1}^{L}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} \frac{N_{ijk}}{N}\log\frac{N_{ijk}}{N_{ij}}\\ &= \frac{1}{N}\text{Score}_{\text{MLC}}(\boldsymbol{G})\end{aligned} \tag{4}$$

In Eq. (4), r_i and q_i denote the number of possible values of y_i and the number of possible value configurations of $\boldsymbol{w}_i$, respectively. y_{ik} and $\boldsymbol{w}_{ij}$ denote the kth value of y_i and the jth configuration of $\boldsymbol{w}_i$. N_{ijk} is the number of instances in which y_i has the kth value and the parents of y_i have the jth configuration. N is the total number of instances in the training dataset, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. In the training dataset, $p(\mathbf{w}_{ij})$ is equal to N_{ij}/N, and $p(y_{ik}, \mathbf{w}_{ij})$ is equal to N_{ijk}/N. Thus, $-\sum_{i=1}^{L} H(y_i|\mathbf{w}_i) = \frac{1}{N}\ \text{Score}_{\text{MLC}}(\boldsymbol{G})$.

Therefore, the problem of Eq. (4) can be solved by using a Bayesian network learning algorithm, which can learn the probabilistic dependency among a set of discrete variables, such as a K2 algorithm. However, the MLC score tends to favor a complete network structure and can cause an overfit problem. To obtain as simple a DAG as possible while optimizing the

```
procedure K2;
inputs:
    $\mathcal{D}$: a dataset containing $L$ labels $(y_1, y_2, \ldots, y_L)$
    $\boldsymbol{s}$: a sequence of the labels
    $p$: an upper bound on the number of parents
outputs: the parents nodes for each node
for $i := 1$ to $L$ do
    $\mathbf{w}_i := \phi$ ;
    $P_{old} := score(\mathbf{w}_i, \mathcal{D})$;
    OKToProceed := true
    while OKToProceed and $|\mathbf{w}_i| < p$ do
        Let $y$ be the node in $Pred(y_i, \boldsymbol{s})$ that maximizes $score(\mathbf{w}_i \cup \{y\}, \mathcal{D})$;
        $P_{new} := score(\mathbf{w}_i \cup \{y\}, \mathcal{D})$;
        if $P_{old} > P_{new}$ then
            $P_{old} := P_{new}$;
            $\mathbf{w}_i := \mathbf{w}_i \cup \{y\}$;
        else OKToProceed := false;
    end {while};
end {for};
end{K2};
```

Fig. 1. Pseudo-code of the K2 algorithm.

MLC score, we add a complexity penalization factor, which results in the minimum description length (MDL) score [9], as shown in Eq. (5). The penalization factor is proportional to the length of describing G.

$$\mathrm{Score}_{\mathrm{MDL}}(\boldsymbol{G}) = \mathrm{Score}_{\mathrm{MLC}}\ (\boldsymbol{G}) - \frac{\log N}{2}\|\boldsymbol{G}\|$$
$$\|\boldsymbol{G}\| = \sum_{i=1}^{L}(r_i - 1)\cdot q_i \tag{5}$$

With this MDL score, a Bayesian network learning algorithm is adopted, and a DAG is built. After finding a DAG, L binary classifiers are learned using the CC concept. Each binary classifier learns from input $\mathbf{x}$ and the predicted outputs for $\mathbf{w}_i$ as additional inputs. The precedent labels $\mathbf{w}_i$ are derived from the DAG.

To automatically build the most probable Bayesian network structure from a dataset, many structure-learning algorithms have been proposed. This paper adopted the score-based K2 algorithm introduced by Cooper and Herskovits [9]. A K2 algorithm is a greedy search algorithm that works as shown in Fig. 1.

As shown in Fig. 1, the K2 algorithm takes an L-dimensional label dataset, a sequence of labels $\boldsymbol{s}$, and a positive integer p. Initially, every label has no parents. The algorithm incrementally adds a label as a parent of the current label if the addition of the label improves the MDL score. In the algorithm, $Pred(y_i, \boldsymbol{s})$ is the set of labels preceding y_i in $\boldsymbol{s}$. The K2 algorithm searches for the parents of y_i in $Pred(y_i, \boldsymbol{s})$. For example, the first label in $\boldsymbol{s}$ becomes a root, and the second label can be the direct successor of the root or it can be another root. If the number of parents of the current label become equal to p, the K2 algorithm stops finding parents for the current node. In this manner, the K2 algorithm finds an approximate DAG very efficiently. Since the K2 algorithm constructs a DAG from a given sequence of labels, the resulting DAG may change if the sequence changes. Fig. 2 shows an example of a DAG constructed by the K2 algorithm.

For the construction of classifiers based on a found DAG, we can adopt several strategies for choosing additional inputs for a label. Since a found DAG can be considered as a Bayesian network, where a node is independent from its ancestors given its parents, we basically use only the parents of a label as the additional inputs for the classifier of the label. We call this the parent set strategy. However, the independence condition may not be guaranteed because we built the DAG heuristically. For better classification performance, we try two more strategies: an ancestor set and a correlated ancestor set.

In the parent set strategy, only the direct precedents of a label are used as additional inputs. In the ancestor set strategy, all the predecessors are used as additional inputs. In the correlated ancestor set strategy, some of the strongly correlated labels in the predecessors are used. The conditional entropy is used for choosing the k most highly correlated predecessors of a label. Let us consider an example of building a DAG, as shown in Fig. 2, with $k = 5$. For a label y_{10}, the parent set is $\{y_3, y_8, y_9\}$; the ancestor set is $\{y_1, y_2, y_3, y_5, y_8, y_9\}$; and the correlated ancestor set is a subset of $\{y_1, y_2, y_3, y_5, y_8, y_9\}$ composed of those highly correlated with y_{10} from the viewpoint of conditional entropies.

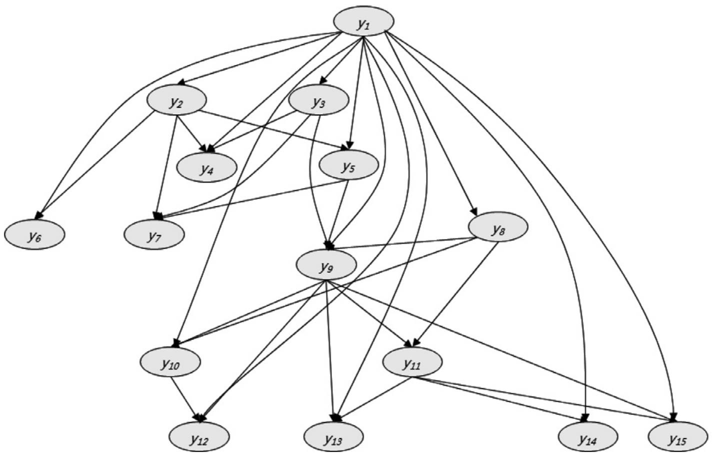

Fig. 2. Example of a directed acyclic graph with $\mathbf{y} = (y_1, y_2, \ldots, y_{15})$.

Table 4
Characteristics of the experimental multi-label dataset.

Dataset	Domain	L	N	D	*LCard*	*LDen*	Unique	*PUniq*	*PMax*
tmc2007	Text	22	28596	500b	2.220	0.101	1172	0.041	0.087
slashdot	Text	22	3782	1079b	1.181	0.054	156	0.041	0.139
genebase	Biology	27	662	1186b	1.252	0.046	32	0.048	0.257
medical	Text	45	978	1449b	1.245	0.028	94	0.096	0.158
enron	Text	53	1702	1001b	3.378	0.064	753	0.442	0.096
langlog	Text	75	1460	1004b	1.180	0.016	304	0.208	0.142
mediamill	Video	101	43907	120n	4.376	0.043	6555	0.149	0.054
bibtex	Text	159	7395	1836b	2.402	0.015	2856	0.386	0.064

4. Experiments

This section provides details the experimental results. Section 4.1 describes the datasets and relevant statistics. Section 4.2 describes the evaluation measures for multi-label learning. In Section 4.3, the proposed multi-label learning methods are compared to the recent work including well-known methods with proven success in multi-label classification. In Section 4.4, since the K2 algorithm constructs a DAG from a given sequence of labels, we also analyze the sensitivity of our methods to the sequences of labels given to the K2 algorithm.

4.1. Datasets

A total of 8 multi-label datasets are used for experiments: *tmc2007*, *slashdot*, *genebase*, *medical*, *enron*, *langlog*, *mediamill*, and *bibtex*. These multi-label datasets were obtained from various domains, such as text classification, biological classification, and multimedia classification. The inputs of datasets are binary or numeric, and the values of all the labels are binary. All of these datasets are publically available at Mulan[1] and Meka.[2] The *tmc2007* dataset has 49,060 features. Therefore, we select the top 500 features using the χ^2 feature-ranking method to reduce computational cost. A similar approach for feature selection has been used in many previous studies [20,23,25].

The associated statistics of the datasets are summarized in Table 4. **L**, **N**, and **d** represent the number of labels, the number of instances, and the number of feature dimensions, respectively. The feature types of datasets, i.e., binary features (b) or numeric features (n), are also indicated together with **d**. Furthermore, various multi-label statistics are also analyzed. The label cardinality (*LCard*) is a standard measure of "multi-labelled-ness" introduced by Tsoumakas et al. [30], as shown in Eq. (6).

$$LCard(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} y_{ij} \tag{6}$$

[1] Available at http://mulan.sourceforge.net/datasets-mlc.html.
[2] Available at http://meka.sourceforge.net/#datasets.

where $\mathcal{D}$ is a dataset, N is the number of instances, L is the number of labels, and y_{ij} is the value of the j-th label of the i-th instance. Since the labels are binary, it is simply the average number of labels per instance, or the average frequency of all the labels. Similarly, the label density (*LDen*) is the normalized *LCard* by the number of labels.

To measure the uniformity of a dataset, Read et al. introduced the proportion of unique label sets (*PUniq*) [22], as shown in Eq. (7). In Table 4, *Unique* indicates the number of unique value configurations of all the labels in a dataset, and *PUniq* is the normalized *Unique* by the number of instances.

$$PUniq(\mathcal{D}) = \frac{|\{\mathbf{y}|\ \exists\ \mathbf{x}\ :(\mathbf{x},\mathbf{y}) \in \mathcal{D}\}|}{N} \tag{7}$$

Finally, Eq. (8) shows the definition of *PMax*, the maximum of value configuration frequencies normalized with the number of data, where $\mathbf{y}$ is a value configuration of all the labels in $\mathcal{D}$ and Count $(\mathbf{y},\ \mathcal{D})$ is the frequency of $\mathbf{y}$ found in dataset $\mathcal{D}$. This equation was introduced by Read et al. and represents the proportion of instances associated with the most frequently occurring label value configuration [23]. For example, a high *PMax* value indicates a skewed distribution of a dataset.

$$PMax(\mathcal{D}) = \max_{\mathbf{y}} \frac{\text{Count}(\mathbf{y},\ \mathcal{D})}{N} \tag{8}$$

As presented in Table 4, the label densities (*LDen*) tend to be extremely low across all datasets. This tendency implies that the datasets are very sparse, which makes the classification problem more difficult. Moreover, while the possible number of label is 2^L, the proportion of unique label sets (*Unique*) tends to be critically lower than the number of possible cases. This implies that there exist many duplicate label sets. For this reason, we need a multi-label learning method that considers the label correlations.

4.2. Evaluation measures

In multi-label experiments, the performance evaluation is more complicated than in the traditional single-label learning because it should include label set-based evaluation as well as each label-based evaluation. In this paper, four popular evaluation measures designed for multi-label learning are adopted: *0/1 Loss*, *H-Loss*, *J-index*, and *F1-macro*. Two of the measures, *0/1 Loss* and *J-index*, are label set-based evaluation measures, and the other two, *H-Loss* and *F1-macro*, are label-based evaluation measures.

Given a multi-label dataset $\mathcal{D} = \{(\mathbf{x_i},\ \mathbf{y_i})|1 \le i \le N\}$, the four evaluation measures are defined as follows. In the definitions, $\mathbf{y_i} = (y_{i1}, y_{i2}, \ldots, y_{iL})$ represents the label configuration or the output of the ith instance, and $\hat{\mathbf{y}}_i$ is the predicted label configuration. Eq. (9) shows *0/1 Loss*, which is also known as the exact match measure. It counts the cases in which the predicted label set $\hat{\mathbf{y}}_i$ exactly matches the original label set $\mathbf{y_i}$.

$$0/1\text{Loss} = 1 - \frac{1}{N}\sum_{i=1}^{N} \mathrm{I}\big(\mathbf{y_i}, \hat{\mathbf{y}}_i\big)$$
$$\text{where } I(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Since this measure can be very harsh in most case, Hamming loss (*H-Loss*) is used as well. *H-Loss* in Eq. (10) indicates how many labels of an instance are misclassified, evaluating more leniently that *0/1 Loss*. A lower value means better performance in both loss measures.

$$H - Loss = 1 - \frac{1}{N \cdot L}\sum_{i=1}^{N}\sum_{j=1}^{L} I\big(y_{ij},\ \hat{y}_{ij}\big) \tag{10}$$

Eq. (11) is the measure of the Jaccard index (*J-Index*) for the multi-label classification. It enables more sensitive measurement of accuracy than the *H-Loss* measure. The *J-index* is calculated for each instance and then averaged across all instances.

$$J - Index = \frac{1}{N}\sum_{i=1}^{N} \frac{\left|\mathbf{y_i} \wedge \hat{\mathbf{y}}_i\right|}{\left|\mathbf{y_i} \vee \hat{\mathbf{y}}_i\right|} \tag{11}$$

The macro averaged F1-measure (*F1-macro*) is also used for evaluation. *F1-macro* is the average of the F1-measure across all labels, as shown in Eq. (12). The F1-measure of a label is the harmonic mean of its precision and recall.

$$F1 - macro = \frac{1}{L}\sum_{j=1}^{L} F1([\hat{y}_{11}, \ldots, \hat{y}_{ij}], [y_{11}, \ldots, y_{ij}]) \tag{12}$$

4.3. Evaluation results

In this section, we compare the proposed methods with various approaches in the recent publications. First, we analyze the performance of the proposed methods with 8 publicly available datasets with 4 evaluation measures, and then check the superiority of the proposed method over other recent methods.

Table 5
Predictive performance in terms of *0/1 Loss* with ranks.

Dataset	Algorithms									
	ECC		RAkEL		IBLR		DAG-A		DAG-C	
tmc2007	0.767	(4)	0.744	(3)	0.797	(5)	0.662	(1)	0.665	(2)
slashdot	0.620	(1)	0.668	(4)	0.887	(5)	0.624	(2)	0.632	(3)
genebase	0.048	(4)	0.036	(3)	0.097	(5)	0.032	(2)	0.029	(1)
medical	0.328	(2)	0.354	(4)	0.632	(5)	0.349	(3)	0.323	(1)
enron	0.873	(1)	0.896	(3)	0.922	(5)	0.894	(2)	0.907	(4)
langlog	0.796	(4)	0.780	(1)	0.856	(5)	0.782	(2)	0.783	(3)
mediamill	0.938	(5)	0.920	(2)	0.922	(4)	0.921	(3)	0.895	(1)
bibtex	0.846	(1)	0.867	(4)	0.942	(5)	0.863	(2)	0.864	(3)
Avg. rank		**2.8**		**3.0**		**4.9**		**2.1**		**2.3**

Table 6
Predictive performance in terms of *H-Loss* with ranks.

Dataset	Algorithms									
	ECC		RAkEL		IBLR		DAG-A		DAG-C	
tmc2007	0.068	(3)	0.068	(3)	0.078	(5)	0.057	(1)	0.057	(2)
slashdot	0.049	(3)	0.053	(4)	0.073	(5)	0.039	(1)	0.039	(1)
genebase	0.002	(4)	0.001	(1)	0.005	(5)	0.001	(1)	0.001	(1)
medical	0.010	(1)	0.011	(3)	0.025	(5)	0.011	(3)	0.010	(1)
enron	0.056	(3)	0.061	(4)	0.067	(5)	0.052	(1)	0.052	(1)
langlog	0.026	(4)	0.019	(1)	0.031	(5)	0.019	(1)	0.019	(1)
mediamill	0.041	(5)	0.032	(1)	0.038	(4)	0.034	(2)	0.034	(2)
bibtex	0.016	(3)	0.017	(4)	0.023	(5)	0.015	(1)	0.015	(1)
Avg. rank		**3.3**		**2.6**		**4.9**		**1.4**		**1.3**

4.4. Performance analysis

We analyze the performance of the proposed methods with 8 publicly available datasets in various aspects by comparing with the previous approaches which are proven success in multi-label classification: ECC [23], RAkEL [20] and IBLR [10]. Instead of re-evaluation of those methods, we refer to the performance evaluations by Read et al. [23]. Our experiments are also performed in accordance with their experimental setting. We adopt the same classifier, the C4.5 decision tree by Quinlan [19], and the same cross-validation method, the 5×2 cross-validation, as well as the same datasets described in Section 4.1 and the same evaluation measures described in Section 4.2.

For the proposed method, the maximum number of parents in the K2 algorithm is set to 3. Since the K2 algorithm needs a sequence of labels, we build a sequence in the order of labels presented in the dataset. We use only the ancestor set and the correlated ancestor set strategies except the parent set strategy because its performances are lower than those of the others, as expected in Section 3. For the correlated ancestor set strategy, 5 labels in the ancestors are chosen. In the results, DAG-A denotes the proposed approach with the ancestor set strategy, and DAG-C with the correlated ancestor set strategy.

For ECC, we use the evaluation results when the number of ensembles m was set to 10; for RAkEL, the results under the configuration that the size of label sets $k = 3$ and the number of models $m = 2L$, where L is the number of labels; and for IBLR, the results when the size of nearest neighbors k was set to 10. Tables 5 to 8 report the detailed predictive results in terms of the four different evaluation measures. In the tables, numbers in parenthesis are the ranks of each method for each data set.

Table 5 shows the predictive performance in terms of *0/1 Loss*. Notice that, for loss measures, a lower value implies a better performance. For example, in the case of *tmc2007*, the *0/1 Loss* of ECC is 0.767, and its rank is 4 among the five methods: ECC, RAkEL, IBLR, DAG-A, and DAG-C. The last row shows the average rank across all datasets. The ranks of ECC are 4, 1, 4, 2, 1, 4, 5, and 1 for the eight datasets, and its average is 2.8.

Based on the results, both proposed methods outperform the other approaches. Our methods are the best and the second best in the average ranking. Furthermore, DAG-A achieves stable performances across all the datasets. It is the first or the second for every dataset.

As shown in Table 6, the proposed methods yield superior predictive performances in terms of *H-Loss*. Especially, it is noteworthy that DAG-A is always superior to the other methods except with the *mediamill* dataset. For *genebase*, every method shows great predictive performance, with very small differences.

Table 7 shows the predictive performance in terms of *J-Index*. DAG-A and DAG-C show competitive performances compared with the ECC approach. Especially, for *tmc2007*, the proposed methods significantly outperform the other methods with large differences. The ECC method shows good performance in terms of the average rank; however, it is unstable, oscillating from the best to the worst.

Table 7
Predictive performance in terms of *J-index* with ranks.

Dataset	Algorithms									
	ECC		RAkEL		IBLR		DAG-A		DAG-C	
tmc2007	0.517	(4)	0.549	(3)	0.479	(5)	0.603	(1)	0.603	(1)
slashdot	0.495	(1)	0.451	(2)	0.241	(5)	0.441	(3)	0.434	(4)
genebase	0.978	(4)	0.982	(3)	0.950	(5)	0.984	(2)	0.986	(1)
medical	0.772	(1)	0.742	(4)	0.480	(5)	0.743	(3)	0.756	(2)
enron	0.452	(1)	0.410	(2)	0.360	(5)	0.409	(3)	0.395	(4)
langlog	0.148	(1)	0.119	(4)	0.008	(5)	0.129	(2)	0.129	(2)
mediamill	0.380	(5)	0.400	(4)	0.426	(1)	0.418	(3)	0.420	(2)
bibtex	0.348	(1)	0.328	(2)	0.161	(5)	0.299	(4)	0.300	(3)
Avg. rank		**2.3**		**3.0**		**4.5**		**2.6**		**2.4**

Table 8
Predictive performance in terms of *F1-macro* with ranks.

Dataset	Algorithms									
	ECC		RAkEL		IBLR		DAG-A		DAG-C	
tmc2007	0.496	(4)	0.577	(3)	0.434	(5)	0.599	(1)	0.597	(2)
slashdot	0.357	(1)	0.344	(4)	0.155	(5)	0.349	(3)	0.353	(2)
genebase	0.749	(4)	0.761	(3)	0.656	(5)	0.778	(2)	0.779	(1)
medical	0.374	(2)	0.369	(3)	0.194	(5)	0.361	(4)	0.379	(1)
enron	0.210	(1)	0.205	(2)	0.143	(5)	0.148	(4)	0.152	(3)
langlog	0.061	(1)	0.051	(3)	0.014	(5)	0.052	(2)	0.050	(4)
mediamill	0.048	(4)	0.042	(5)	0.179	(3)	0.204	(1)	0.194	(2)
bibtex	0.337	(1)	0.334	(2)	0.139	(5)	0.297	(3)	0.278	(4)
Avg. rank		**2.3**		**3.1**		**4.8**		**2.5**		**2.4**

Table 9
Average rank for each dataset.

Dataset	# Labels	Algorithms				
		ECC	RAKEL	IBLR	DAG-A	DAG-C
tmc2007	22	3.8	3.0	5.0	1.0	1.8
slashdot	22	1.5	3.5	5.0	2.3	2.5
genebase	27	4.0	2.5	5.0	1.8	1.0
medical	45	1.5	3.5	5.0	3.3	1.3
enron	53	1.5	2.8	5.0	2.5	3.0
langlog	75	2.5	2.3	5.0	1.8	2.5
mediamill	101	4.8	3.0	3.0	2.3	1.8
bibtex	159	1.5	3.0	5.0	2.5	2.8
Avg. rank		**2.6**	**2.9**	**4.8**	**2.2**	**2.1**

As shown in Table 8, both proposed methods show very similar results to the ECC approach in terms of *F1-macro*. It is interesting to note that every algorithm performs poorly with the *langlog* data. This is related to the fact that *langlog* is a class-imbalanced dataset, which has low $LDen = 0.016$ and low $PUniqe = 0.208$ but high $PMax = 0.142$ (see Table 4).

To analyze the performance with respect to the number of labels, we observe the average ranks for each dataset, as shown in Table 9. For example, the average rank of the ECC approach for the *tmc2007* dataset is 3.8, which is the average in

Table 10
The win/tie/loss results against the proposed method DAG-C.

Evaluation measures	Algorithms		
	ECC	RAkEL	IBLR
0/1 Loss	5 / 0 / 3	6 / 2 / 0	8 / 0 / 0
H-Loss	5 / 3 / 0	4 / 3 / 1	8 / 0 / 0
J-index	3 / 0 / 5	5 / 0 / 3	7 / 0 / 1
F1-macro	4 / 0 / 4	5 / 1 / 2	8 / 0 / 0
In total	**17 / 3 / 12**	**20 / 4 / 8**	**31 / 0 / 1**

four measures: *0/1 Loss*, *H-Loss*, *J-index*, and *F1-macro*. Note that the proposed methods show good performance regardless of the number of labels in the datasets.

Comparing the performance of DAG-A and DAG-C, both achieve good performance in most cases, and DAG-A is slightly inferior to DAG-C, but there is no significant difference in any dataset. In Table 9, the overall average rank of DAG-A is 2.2 while that of DAG-C is 2.1. However, DAG-A is computationally more expensive than DAG-C because DAG-A uses all the ancestor labels as the additional inputs, whereas DAG-C uses only some of the ancestor labels. Therefore, we conclude that DAG-C is better than DAG-A.

Table 11
The win/tie/loss results between state-of-art multi-label algorithms and DAG-C.

Evaluation measures	Algorithms			
	SSRKDA	*MMFS*	*OOCC*	*GA-PartCC*
J-index	1 / 0 / 2	3 / 0 / 1	1 / 0 / 2	1 / 1 / 2
F1-macro	2 / 0 / 1	–	–	–
H-Loss	1 / 2 / 0	3 / 0 / 1	2 / 1 / 0	1 / 3 / 0
In total	**4 / 2 / 3**	**6 / 0 / 2**	**3 / 1 / 2**	**2 / 4 / 2**

Table 12
Predictive results with random orders.

0/1 Loss dataset	Given order	Random order		
		Avg.	Best	Worst
tmc2007	0.665	0.662	0.661	0.663
slashdot	0.632	0.626	0.622	0.632
genebase	0.029	0.028	0.026	0.029
medical	0.323	0.337	0.329	0.346
enron	0.907	0.895	0.890	0.899
langlog	0.783	0.785	0.782	0.788
mediamill	0.895	0.921	0.919	0.922
bibtex	0.864	0.863	0.861	0.865
H-Loss dataset	Given order	Random order		
		Avg.	Best	Worst
tmc2007	0.057	0.057	0.057	0.057
slashdot	0.039	0.039	0.039	0.040
genebase	0.001	0.001	0.001	0.001
medical	0.010	0.010	0.010	0.010
enron	0.052	0.052	0.052	0.052
langlog	0.019	0.019	0.018	0.019
mediamill	0.034	0.034	0.034	0.034
bibtex	0.015	0.015	0.015	0.015
J-Index dataset	Given order	Random order		
		Avg.	Best	Worst
tmc2007	0.603	0.604	0.605	0.603
slashdot	0.434	0.438	0.441	0.434
genebase	0.986	0.987	0.988	0.986
medical	0.756	0.750	0.759	0.743
enron	0.395	0.404	0.409	0.401
langlog	0.129	0.130	0.137	0.125
mediamill	0.420	0.418	0.420	0.417
bibtex	0.300	0.302	0.304	0.300
F-macro dataset	Given order	Random order		
		Avg.	Best	Worst
tmc2007	0.597	0.597	0.600	0.595
slashdot	0.353	0.345	0.350	0.337
genebase	0.779	0.789	0.798	0.781
medical	0.379	0.382	0.385	0.375
enron	0.152	0.153	0.160	0.145
langlog	0.050	0.053	0.054	0.052
mediamill	0.194	0.200	0.203	0.196
bibtex	0.278	0.280	0.280	0.278

Finally, we analyze the win/tie/loss results for DAG-C against the three compared approaches, as shown in Table 10. When a difference between two evaluation results is less than or equal to 0.001, it is counted as a tie. The proposed method DAG-C has the same or more wins than ties, except for ECC with *J-index*. Also, DAG-C has just one loss in all the cases for the *H-Loss* measure, which is the most standard measure for measuring the accuracy of predictive performance.

4.5. Comparison with other recent work

We also compare the performance of DAG-C to various recent multi-label classification researches. We choose the recent researches which share at least three common datasets and at least two common evaluation measures for performance evaluation with our own. Finally, *SSRKDA* [27], *MMFS* [14], *OOCC* [25], and *GA-PARTCC* [11] are selected and their performances are quoted from their original work. Since the evaluation results are quoted from multiple previous work, different validation techniques were adopted in their work. Some of the researches used five-fold cross-validation techniques while the other researches used three-fold or leave-one-out cross-validation techniques. Since the variance in cross-validations is theoretically independent from the value of *k* [13], we analyze the competitiveness of our approach with those recent researches. Most of the researches did not use *0/1 Loss* as the evaluation measure, so we compare with only the other 3 evaluation measures: *H-Loss*, *J-index* and *F1-macro*.

We analyze the win/tie/loss scores for DAG-C against the recent multi-label researches in Table 11. When a difference in performances of two approaches is less than or equal to 0.001, it is counted as a tie. For example, in Table 11, DAG-C

Table 13
Predictive results according to different score metrics.

0/1 Loss dataset	Score metrics				
	MDL	AIC	K2	BAYES	BDeu
tmc2007	0.663	0.649	0.657	0.655	0.654
slashdot	0.621	0.626	0.624	0.632	0.633
genebase	0.030	0.032	0.032	0.030	0.030
medical	0.316	0.344	0.348	0.352	0.332
enron	0.858	0.889	0.905	0.890	0.892
langlog	0.784	0.788	0.779	0.790	0.788
mediamill	0.892	0.888	0.920	0.886	0.887
bibtex	0.860	0.858	0.869	0.860	0.857
H-Loss dataset	Score metrics				
	MDL	AIC	K2	BAYES	BDeu
tmc2007	0.057	0.057	0.057	0.058	0.058
slashdot	0.039	0.039	0.039	0.040	0.040
genebase	0.001	0.001	0.001	0.001	0.001
medical	0.010	0.010	0.010	0.011	0.010
enron	0.050	0.054	0.052	0.055	0.053
langlog	0.018	0.018	0.018	0.018	0.019
mediamill	0.035	0.034	0.034	0.034	0.034
Bibtex	0.015	0.015	0.015	0.015	0.015
J-Index dataset	Score metrics				
	MDL	AIC	K2	BAYES	BDeu
tmc2007	0.603	0.602	0.605	0.600	0.601
slashdot	0.442	0.437	0.442	0.432	0.431
genebase	0.985	0.983	0.984	0.985	0.985
medical	0.765	0.751	0.750	0.745	0.756
enron	0.442	0.397	0.402	0.400	0.395
langlog	0.131	0.125	0.134	0.128	0.125
mediamill	0.417	0.420	0.418	0.423	0.419
bibtex	0.297	0.297	0.296	0.295	0.303
F-macro dataset	Score metrics				
	MDL	AIC	K2	BAYES	BDeu
tmc2007	0.601	0.599	0.598	0.591	0.594
slashdot	0.342	0.340	0.345	0.344	0.339
genebase	0.764	0.779	0.751	0.766	0.780
medical	0.393	0.373	0.378	0.377	0.385
enron	0.167	0.152	0.151	0.150	0.150
langlog	0.057	0.059	0.055	0.059	0.049
mediamill	0.192	0.188	0.206	0.189	0.190
Bibtex	0.275	0.275	0.277	0.272	0.275

is compared with *SSRKDA* method at the second column. The win/tie/loss score with *J-index* is 1/0/2, which means that DAG-C wins at 1 datasets, ties at 0 dataset and loses at 2 dataset. DAG-C wins scores 2 wins and 1 lose with *F1-macro* and 1 wins and 2 ties with *H-Loss*. Therefore, the total scores of DAG-C against *SSRKDA* is 4/2/3. A dash in cells means that the performance evaluation with the measure is not available.

As shown in the table, DAG-C wins against 3 methods, ties with 1 method and does not lose to any methods. We can say that the results strongly support the competitiveness of the proposed methods compared to the recent researches.

4.6. Discussion of building a DAG

To build a DAG, we use the K2 algorithm presented in Section 3. This algorithm requires a given sequence of labels, which has a great effect on the structure of the network. Thus, the final structure of a DAG depends on the sequence of labels given to the K2 algorithm. We give labels to the K2 algorithm in the order of the labels in each dataset. In this section, we empirically analyze the classification performance with respect to the sequence of labels given to the K2 algorithm. We build a DAG with random sequences of labels using the MDL score, and our DAG-C approach is used to evaluate the results. We repeat the experiment five times. Table 12 shows the results across 8 datasets. In Table 12, the columns for *Avg.*, *Best*, and *Worst* show the average, best, and worst performance of the five experiments with random sequences. We find that the predictive results do not much depend on the sequences of labels because the predictive performances are very similar in the datasets, which indicates that our approach is robust to the sequence of labels given to the K2 algorithm.

We used the MDL score metric to build a DAG with the K2 algorithm. However, in Bayesian network learning research, various score metrics have been proposed aiming to find the network that best fits a dataset [5]. We also empirically analyze the performance of our DAG-C approach with respect to the score metrics. We perform the comparative experiments with four score metrics: AIC, K2, BAYES, and BDeu, which are widely used for Bayesian network learning. Table 13 shows the comparative results across the 8 datasets. The results show no significant differences among the predictive results of each score metric. Therefore, we conclude that any score metric will work in our approach.

5. Conclusions

This paper proposed a novel multi-label learning method that considers the correlations of labels explicitly while maintaining acceptable computational complexity. Starting from the simple intuitive notion of measuring the correlation between labels in a directed acyclic graph (DAG), the proposed method sought the DAG structure that maximized the correlations among labels. We quantified the correlation among labels with the conditional entropy and found a DAG that maximized the sum of conditional entropies between all parents and children nodes. Thus, the highly correlated labels could be sequentially ordered in chains obtained from the DAG, and the predictive power could be maximized by utilizing a CC approach with these chains. Because building a DAG is equivalent to learning a Bayesian network, the K2 algorithm was used to build a DAG that is widely used for Bayesian network learning.

The proposed methods were compared with state-of-the-art approaches. Experimental results on 8 multi-label datasets showed that the proposed method outperformed the other approaches. Due to the competitive accuracy and high cost-efficiency, the proposed method is expected to be a practical and attractive approach in multi-label learning.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2014M3C4A7030503). Also, this research was supported by ICT R&D program of MSIP/IITP (10041244, Smart TV 2.0 Software Platform).

References

[1] G. Boella, L. Di Caro, D. Rispoli, L. Robaldo, A system for classifying multi-label text into EuroVoc, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, 2013, pp. 239–240.
[2] R.R. Bouckaert, Probabilistic network construction using the minimum description length principle, in: Symbolic and Quantitative Approaches to Reasoning and Uncertainty, 1993, pp. 41–48.
[3] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[4] M. Boutell, X. Shen, J. Luo, C. Brown, Multi-label semantic scene classification, Technical Report, Dept. Computer Science, Univ. Rochester, 2003.
[5] A.M. Carvalho, Scoring functions for learning bayesian networks, Inesc-id Tec. Rep. (2009).
[6] W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Mach. Learn. 76 (2–3) (2009) 211–225.
[7] W. Cheng, E. Hüllermeier, K.J. Dembczynski, Bayes optimal multilabel classification via probabilistic classifier chains, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 279–286.
[8] E.A. Cherman, M.C. Monard, J. Metz, Multi-label problem transformation methods: a case study, CLEI Electron. J. 14 (1) (2011) 1–10.
[9] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Mach. Learn. 9 (4) (1992) 309–347.
[10] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Advances in Knowledge Discovery and Data Mining, 2004, pp. 22–30.
[11] E.C. Gonçalves, A. Plastino, A.A. Freitas, Simpler is Better: a novel genetic algorithm to induce compact multi-label chain classifiers, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, 2015, pp. 559–566.
[12] T. Grigorios, K. Ioannis, Multi label classification: an overview, Int. J. Data Warehous. Min. 3 (3) (2007) 1–13.

[13] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: The International Joint Conference on Artificial Intelligence (IJCAI), 1995, pp. 1137–1145.
[14] J. Lee, D.W. Kim, Memetic feature selection algorithm for multi-label classification, Inf. Sci. 293 (2015) 80–96.
[15] X. Li, F. Zhao, Y. Guo, Conditional restricted Boltzmann machines for multi-label learning with incomplete labels, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, 2015, pp. 635–643.
[16] H.Y. Lo, S.D. Lin, H.M. Wang, Generalized k-labelsets ensemble for multi-label and cost-sensitive classification, IEEE Trans. Knowl. Data Eng. 26 (7) (2014) 1679–1691.
[17] Y. Luo, D. Tao, B. Geng, C. Xu, S.J. Maybank, Manifold regularized multitask learning for semi-supervised multilabel image classification, IEEE Trans. Image Process. 22 (2) (2013) 523–536.
[18] G. Madjarov, D. Kocev, D. Gjorgjevikj, S. Džeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognit. 45 (9) (2012) 3084–3104.
[19] J.R. Quinlan, C4.5: Programs for Machine Learning, Elsevier, 2014.
[20] J. Read, B. Pfahringer, G. Holmes, Multi-label classification using ensembles of pruned sets, in: Eighth IEEE International Conference on Data Mining, 2008, ICDM'08, 2008, pp. 995–1000.
[21] J. Read, A. Puurula, A. Bifet, Multi-label classification with meta-labels, in: IEEE International Conference on Data Mining (ICDM), 2014, pp. 941–946.
[22] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: Proceedings of the European Conference on Machine Learning, 2009, pp. 254–269.
[23] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2011) 333–359.
[24] R.E. Schapire, Y. Singer, BoosTexter: a boosting-based system for text categorization, Mach. Learn. 39 (2) (2000) 135–168.
[25] P.N. Silva, E.C. Gonçalves, A. Plastino, A.A. Freitas, Distinct chains for different instances: an effective strategy for multi-label classifier chains, in: Machine Learning and Knowledge Discovery in Databases, 2014, pp. 453–468.
[26] F. Sun, J. Tang, H. Li, G.J. Qi, T.S. Huang, Multi-label image categorization with sparse factor representation, IEEE Trans. Image Process. 23 (3) (2014) 1028–1037.
[27] M.A. Tahir, J. Kittler, A. Bouridane, Multi-label classification using stacked spectral kernel discriminant analysis, Neurocomputing 171 (2016) 127–137.
[28] F. Tai, H.T. Lin, Multilabel classification with principal label space transformation, Neural Comput. 24 (9) (2012) 2508–2542.
[29] G. Tsoumakas, I. Katakis, I. Vlahavas, Effective and efficient multilabel classification in domains with large number of labels, in: Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD), 2008, pp. 30–44.
[30] G. Tsoumakas, I. Katakis, I. Vlahavas, Random k-labelsets for multilabel classification, IEEE Trans. Knowl. Data Eng. 23 (7) (2011) 1079–1089.
[31] D. Vasisht, A. Damianou, M. Varma, A. Kapoor, Active learning for sparse bayesian multilabel classification, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 472–481.
[32] M.L. Zhang, K. Zhang, Multi-label learning by exploiting label dependency, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 999–1008.