ML-FOREST: A Multi-Label Tree Ensemble Method for Multi-Label Classification

Qingyao Wu, Mingkui Tan, Hengjie Song, Jian Chen, and Michael K. Ng

Abstract—Multi-label classification deals with the problem where each example is associated with multiple class labels. Since the labels are often dependent to other labels, exploiting label dependencies can significantly improve the multi-label classification performance. The label dependency in existing studies is often given as prior knowledge or learned from the labels only. However, in many real applications, such prior knowledge may not be available, or labeled information might be very limited. In this paper, we propose a new algorithm, called ML-FOREST, to learn an ensemble of hierarchical multi-label classifier trees to reveal the intrinsic label dependencies. In ML-FOREST, we construct a set of hierarchical trees, and develop a label transfer mechanism to identify the multiple relevant labels in a hierarchical way. In general, the relevant labels at higher levels of the trees capture more discriminable label concepts, and they will be transferred into lower level children nodes that are harder to discriminate. The relevant labels in the hierarchy are then aggregated to compute label dependency and make the final prediction. Our empirical study shows encouraging results of the proposed algorithm in comparison with the state-of-the-art multi-label classification algorithms under Friedman test and post-hoc Nemenyi test.

Index Terms—Multi-label classification, label dependency, label transfer, tree classifier, ensemble methods

1 INTRODUCTION

MULTI-LABEL classification aims to predict the presence or absence of certain labels of an example which is associated with multiple classes. Different from classical multi-class problems, where an example is associated with only one single label, multi-label classification is more general since real-world objects often contain multiple semantic objects. For example, a real-world image usually belongs to multiple categories based on different contexts, such as water, ship, etc.; while a text document can be classified into a set of topics, such as *news*, *sports*, etc. In the last decades, multi-label classification problem has received broad attention from various research domains, such as text categorization [1], [2], [3], bioinformatics [4], [5], [6], and computer vision [7], [8], [9].

A straightforward multi-label classification approach is *binary relevance* (BR) [10], which decomposes the problem into a set of single-label multi-class problems. In this way, a set of multi-class classifiers are learnt and then applied to do prediction. This simple method, however, totally neglects the dependencies among multiple labels. In practice, multiple objects in an example (such as an image) may have strong relations or dependencies. For example, if a *ship* category is presented in an image, it is very likely that the *water* category is also in that image. Exploiting such label dependency may significantly improve the prediction performance for multi-label classification.

Plenty of previous studies [11], [12] have tried to exploit the label dependency to improve the prediction performance. However, how to effectively model the label dependency explicitly is still a challenging problem. In [13], the authors simply assume that the label dependency is provided as prior knowledge in real-world situations. Some other approaches, like [14], [15], consider learning the label dependency from very limited information, e.g., the co-occurrence of the labels in the training set. However, these kind of learning methods may cause over-fitting issues [11].

In this paper, we propose a new tree ensemble algorithm, called *ML-FOREST*, to explicitly exploit label dependency for multi-label classification. In ML-FOREST, a set of hierarchical trees are constructed to learn the label dependency, and then combined as an ensemble to do multi-label prediction. Specifically, the primary focus of this paper is to find a good hierarchical structure so that two relevant instances with strong label dependency will be located in the same node of the tree. To achieve this, we design a new tree generation algorithm to partition the learning data into smaller subsets from the root to the leaves, and then identify relevant labels for each node with a label transfer mechanism.

For the first task, we train multi-class classifiers at each node to divide the data into child nodes. Here, each data instance is partitioned into one child node according to the classifier prediction results, and the class label with highest probability given at the node is considered as its relevant label.

For the second task of the algorithm, a label transfer mechanism is involved to recursively propagate the relevant labels from the root down to the leaf nodes. For example, if a relevant label is found at a node, all of its children nodes would automatically belong to this relevant label, and we seek a new

- *Q. Wu, M. Tan, H. Song, and J. Chen are with the School of Software Engineering, South China University of Technology, Guangzhou, Guangdong 510641, China. E-mail: {qyw, sehjsong, ellachen}@scut.edu.cn, tanmingkui@gmail.com.*
- *M. K. Ng is with the Mathematics Department, Hong Kong Baptist University, Hong Kong. E-mail: mng@math.hkbu.edu.hk.*

Manuscript received 5 Aug. 2015; revised 28 May 2016; accepted 10 June 2016. Date of publication 15 June 2016; date of current version 7 Sept. 2016. Recommended for acceptance by J. Gama.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2581161

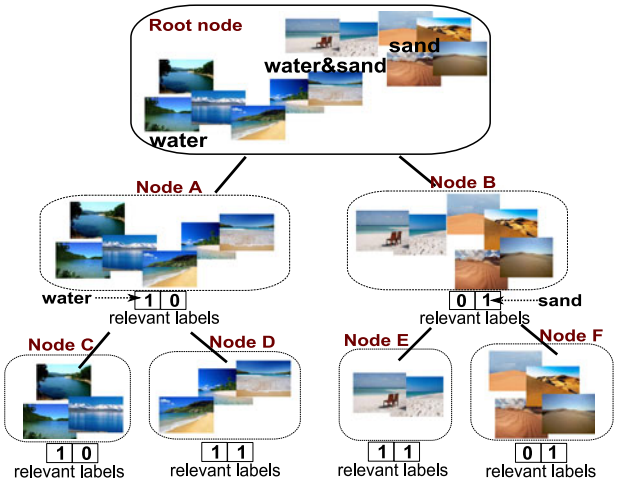

Fig. 1. An example of a hierarchical tree for scene classification.

relevant label (if any) which respects the label dependencies of the instances in children nodes. In the end, each leaf node is characterized by multiple relevant labels given by the nodes at different levels of the tree. This leads to a new label dependency representation, where the learning models at different levels work together effectively to reveal multiple label concepts belonging to the given data. Intuitively, the relevant labels at high levels in the hierarchy may tend to capture "more significant" label concepts and hence are thematically more general; while the relevant labels at low levels would capture "less significant" label concepts and hence are thematically more specific.

In Fig. 1 we illustrate the above ideas by showing an example of a hierarchical tree constructed in a multi-label scene classification task. In this figure, the instances at the root node belong to *water* or *sand* or both these two classes simultaneously (e.g., a beach scene contains both water and sand). A hierarchical tree then is constructed to partition the data from the root to the leaf nodes, and we identify the relevant labels at each node to capture the label concepts based on the label transfer mechanism. In particular, the data instances are partitioned into the same node if they are close to each other and they would be labeled hierarchically such that the label concepts at higher layers are often more easier to be characterized.

In the above example, part of the beach images would be partitioned into node **A** and labeled as *water* at the first layer of the tree; while the rest beach images would be labeled as *sand* in node **B**, depending on which class the images belong to. The resulting class decision hyperplanes will further split the data of node **A** into nodes **C** and **D**. It is worth mentioning that the *water* class associated to node **D** is inherited from node **A**, whilst the sand class associated to node E is given by node **B**. For convenience, hereafter this hierarchical multi-label tree is referred to (ML-TREE).

The major contributions of this paper are as follows:

- We propose a new hierarchical tree algorithm, called ML-TREE, to solve the multi-label classification problem. Unlike BR method which transforms a multi-label classification problem into independent binary classification problems, our algorithm exploits the intrinsic label dependency of the data and incorporates the ML-TREE structure to find the relevant labels of an instance with multiple labels. Therefore, the proposed approach provides a principled way for modeling the intrinsic label dependency of the data into a tree structure.
- We design a label transfer mechanism to find the relevant labels in the hierarchy. The labels of high levels in the hierarchy will be used as priors for the nodes in low levels to reduce the label space. Therefore, building the classifier model for low level labels can be very efficient.
- We develop an ensemble strategy to construct multiple hierarchical multi-label trees and combine the predictions of different trees as an ensemble to make predictions.
- We evaluate the empirical performance by conducting an extensive set of experiments on real-world problems in text classification, computer vision and bioinformatics. Experimental results demonstrate that the ML-FOREST approach is highly competitive to the state-of-the-art approaches under Friedman and Nemenyi tests [16].

The rest of this paper is organized as follows. The problem of multi-label classification and related work are introduced in Section 2. The proposed methodology is then described in Section 3. The data sets, the experimental setup and experimental results are discussed in Section 4. Finally, conclusions are drawn in Section 5.

2 Related Work

Let $\mathcal{X} = \mathcal{R}^d$ be the d-dimensional input space. Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathcal{X}$ contains m instances and $\mathbf{y}_i \in \mathcal{Y} = \{0, 1\}^q$ consists of q possible labels, the multi-label learning aims to learn a hypothesis $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an input $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$. Regarding the label $\mathbf{y}_i$ of the ith example, $y_i^j = 1$ if x_i contains the jth target, and $y_i^j = 0$ otherwise.

In the past decades, a number of multi-label classification approaches have been developed regarding various areas, such as text categorization [1], [2], [3], bioinformatics [4], [5], [6], and computer vision [7], [8], [9]. These works have revealed that exploiting the dependency among different labels is crucial to improve the performance for multi-label classification. For example, Zhang et al. [14] summarized the existing multi-label approaches into three categories based on the orders of dependencies exploited in the system, including *First-order* approaches, *Second-order* approaches and *Higher-order* approaches.

First-order approaches decompose the multi-label classification task into a number of independent tasks [17], [18]. The most common method is the *binary relevance* method [10], which transforms a multi-label problem into multiple separate and independent binary problems, one for each label. It is clear that the first-order methods are incapable of label dependency, which might cause a degradation of the predictive performance.

Second-order approaches consider the pairwise relations between labels, such as the interaction between any pair of labels [19], [20]. In general, such pairwise label dependency is estimated by the co-occurrence or some other equivalent

measures of the labels. However, these approaches might over-fit the training data since these dependencies are usually inaccurate.

High-order approaches consider even higher order of relations among labels, such as the *full-order* style imposed on all the labels [21], [22], [23], [24]. For example, Clare and King [25] applied the dependencies between all labels to enhance the multi-label classification performance. However, the *full-order* approaches are usually impossible when the number of labels is large, where the number of possible label subset combinations could be exponentially huge.

Hierarchical tree-based model is a family of learning algorithms with simple theoretical foundation, and has been widely applied in multi-label classification [22], [25], [26], [27], [28], [29], [30]. For instance, Clare and King [25] adapted the C4.5 algorithm for multi-label data by modifying the formula of entropy calculation. Comité et al. [31] learned multi-label alternating decision trees from text data. Blockeel et al. [32] proposed predictive clustering trees (PCT) to make multi-target prediction/multi-label classification; while Vens et al. [23] introduced a high-order approach to extend PCT algorithm where these classes are organized in a hierarchical form, but the hierarchical label dependencies should be provided by external information as prior knowledge. Tsoumakas et al. [22] proposed a hierarchy of multi-label classifiers (HOMER) algorithm using a tree structure to handle problems with a large number of labels, in which the whole label set is disjointed into subsets to construct the tree by using a balanced clustering algorithm. This method is a high-order approach and it does not require the label structure as prior knowledge, but it is computationally inefficent to fine-tuning the parameters involved in constructing the hierarchical model.

Recently, various researchers (see [33], [34], [35]) have exploited the random forest type ensemble methods to enhance the learning performance. Motivated by recent progress in ensemble learning, we propose to exploit label dependencies to improve the multi-label prediction performance via the ensemble of hierarchical trees, namely ML-FOREST. Our proposed ML-FOREST method is a *high-order* method, where each classifier tree addresses dependency among a subset of labels based on relevant labels generated from the root to the leaves. Note that in this way, the size of the label subset is much reduced when considering their dependencies, and such intrinsic label dependencies will be explicitly presented in the hierarchical trees. More importantly, the learned dependency which offers a natural way to gain more insights into the multi-label classification, will lead to improvement in predictive performance and lower computational cost compared to other state-of-the-art tree based multi-label learning algorithms.

The proposed ML-FOREST algorithm is different from PCT [32], HOMER [22], and Two Stage Architecture (TSA) [36].

- In PCT [32], a variance function is employed to split the learning data by maximizing the cluster homogeneity, and a prototype function is used to compute a label for each leaf. In our proposed method, classifier models are constructed to partition the data into child nodes; while we identify the relevant labels at each node and transfer the labels from the root to leaves in a top-down manner to preserve label dependence in the hierarchy.
- HOMER starts with a root node containing all the possible classes, followed by a recursive process to partition the classes into the leaves (each class corresponds to one leaf). Each internal node contains the union of the labels of its children. On the contrary, our proposed algorithm recursively partitions learning data into child nodes, in which every internal node consists of all instances of its children. In addition, for a multi-label instance $\mathbf{x}$, HOMER forwards $\mathbf{x}$ into multiple leaves, and the union of the single-labels in the corresponding leaves is used as the multi-label output of the HOMER approach; while in our proposed algorithm, the multi-label instance $\mathbf{x}$ is forwarded into only one leaf node, and the corresponding labels of nodes in the path from the root to this leaf are taken as the multi-label output of the proposed approach.
- Madjarov et al. [36] propose a Two Stage Architecture algorithm for multi-label learning. The algorithm is implemented by using two layers. In particular, binary relevance models are built in the first layer to reduce the complexity of the training of pair-wise models in the second layer. This method is a second-order approach. While in our proposed algorithm, we construct a hierarchical tree structure, which models the label dependency following the divide-and-conquer paradigm. We use a recursive process to partition the data into smaller subsets, and this process continues until the remaining instances at the node cannot be further split by the induced classifier. Therefore, our algorithm is a high-order approach that constructs multiple-layer models.

3 METHODOLOGY

In this section, we present the ML-FOREST method in details. We first describe the classifier tree construction algorithm as well as the label transfer mechanism for exploiting label dependencies. Next, we incorporate the classifier trees into a forest via a new ensemble framework to further improve the prediction performance, and give the computational complexity of the proposed algorithm.

3.1 The ML-TREE Algorithm

Statistically, the label dependency can be categorized into two groups, namely conditional and unconditional dependency. Here the conditional label dependency captures the dependency of the labels given a specific instance $\mathbf{x} \in \mathcal{X}$; while the unconditional label dependency is the expected dependency averaged over the marginal distribution of all instances [12].

The joint conditional probability distribution $p(\mathbf{y}|\mathbf{x})$, specifies the probability of the label combination for a specific instance, which provides a convenient point of

departure for analyzing the conditional label dependence. Mathematically, $p(\mathbf{y}|\mathbf{x})$ can be written as

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= p(y^1|\mathbf{x})p(y^2,\ldots,y^q|y^1,\mathbf{x}) \\ &= p(y^1|\mathbf{x})p(y^2|y^1,\mathbf{x})p(y^3,\ldots,y^q|y^1,y^2,\mathbf{x}) \\ &= p(y^1|\mathbf{x})p(y^2|y^1,\mathbf{x})\ldots p(y^q|y^1,\ldots,y^{q-1},\mathbf{x}), \end{aligned} \tag{1}$$

where the elements of $\mathbf{y} = (y^1, y^2, \ldots, y^q)$ can be arranged by arbitrary order. Based on the above formulation, the joint conditional probability $p(\mathbf{y}|\mathbf{x})$ can be estimated by multiple steps, with each step for one class. To be more specific, we can build a model to compute a prediction probability for one label given $\mathbf{x}$ (e.g., $p(y^1|\mathbf{x})$), then the output is used as the prior to help the estimation of the probability for another label (e.g., $p(y^2|y^1,\mathbf{x})$).

Recently, researchers have considered the classifier chain (CC) idea (see [37], [38]) to model the underlying label dependency. The classifier chain [38] algorithm which can be considered as a deterministic approximation of probability only using $\{0,1\}$ values [37]. All these two algorithms build q classifiers for estimation w.r.t. q class labels, in which the jth classifier is used to estimate $p(y^j = 1|\mathbf{x}, y^1, \ldots, y^{j-1})$ and the result is further propagated to the $(j+1)$th classifier by constructing a new feature vector augmented by the value of jth label. The $(j+1)$th classifier is to estimate $p(y^{j+1} = 1|\mathbf{x}, y^1, \ldots, y^j)$ and generates a new feature vector $(\mathbf{x}, y^1, \ldots, y^{j+1})$ for the next classifier. The order of labels to be computed in PCC and CC has a great impact on the classification performance, and how to determine the order of labels is still an open question [38].

Motivated by recent progress in exploiting label dependency, in this paper, we propose a new hierarchical tree algorithm, called ML-TREE, which explicitly considers the intrinsic label dependency in a hierarchy way. The pseudo-code of the algorithms is described in Algorithm 1 and 2. Specifically, there are three folds constructing the hierarchical structure: 1) At each internal node, a multi-class classifier model is built to partition the training data into smaller subsets according to the predictions of the model; 2) A set of relevant labels are identified at each node for multi-label classification; 3) The relevant labels of a node in high levels will be transferred into its child nodes, which consider the remained labels according to the classifier built in this phase.

Algorithm 1. ML-TREE

Input: A training data set $\mathcal{D}$, and a relevant label vector $\mathbf{b}$ = none
Output: A hierarchical multi-label tree
1: ($\mathbf{b}$, h, $\mathcal{P}$) = SPLITTEST($\mathcal{D}$, $\mathbf{b}$)
2: **if** $h \neq$ none $\wedge$ Acceptable($\mathcal{P}$) **then**
3: **for** $D_i \in \mathcal{P}$ **do**
4: $tree_i$=ML-TREE($\mathcal{D}_i$, $\mathbf{b}$)
5: **end for**
6: **return** node(h, $\mathbf{b}$, $\cup_i\{tree_i\}$)
7: **else**
8: **return** leaf(h, $\mathbf{b}$)
9: **end if**

Algorithm 2. SPLITTEST

Input: A training data set $\mathcal{D}$, a relevant label vector $\mathbf{b}_p$ from parent
Output: A classifier h, a new relevant label vector $\mathbf{b}$, and a partition $\mathcal{P}$ for current node
1: compute $\mathbf{p}$ using Eq. (2)
2: compute $\mathbf{b}$ using Eq. (3) and (4)
3: (h, $\mathcal{P}$) = (none, ϕ)
4: h = build classifier on $\mathcal{D}$ for those labels which have not been identified according to $\mathbf{b}$
5: **if** $h \neq$none **then**
6: $\mathcal{P}$= partition $\mathcal{D}$ using h
7: **end if**
8: **return** ($\mathbf{b}$, h, $\mathcal{P}$)

We seek to find the relevant labels to be associated with the data examples at each node. If no examples are found, the majority class of its parent node is returned; otherwise, the majority class of all examples at the node is returned. After that, the relevant labels identified by the nodes at higher levels are transferred into the nodes at lower levels as prior label information. This major step of ML-TREE, i.e., SPLITTEST, is detailed as follows:

i) The ML-TREE function invokes itself recursively for each partition using the SPLITTEST function to train a multi-class classifier and identify the relevant labels. At each node splitting, SPLITTEST builds a group of one-against-all binary classifiers [39] for those remained labels that have not been identified in any of its parent nodes, i.e., the labels with $b^j = 0$ (see Line 4 in Algorithm 2) .

Then, each example is classified into one class with maximum confidence score from the multi-class classifier, and it is partitioned into the corresponding child node in next layer (see Line 6 in Algorithm 2). Note that it is possible that the confidence scores of two (or more) classifiers might be equally maximal. In this case, the example is classified into the class with the largest prior.

ii) In order to find the relevant labels for each node, we design a label purity vector, denoted by $\mathbf{p} = [p^1, \ldots, p^q]^\top$, to represent the purities of different classes. Specifically, we calculate each class label's data purity by

$$p^j = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}_i \in \mathcal{D}} y_i^j, \tag{2}$$

where $p^j \in [0,1]$ is the purity for the jth class label, $\mathcal{D}$ is the examples at the node, and $|\mathcal{D}|$ is the number of examples in $\mathcal{D}$.

We then construct a relevant label vector, $\mathbf{b} = [b^1, \ldots, b^q]^\top$, and incorporate the purities to seek the majority labels as the relevant labels of a node by

$$b^j = \begin{cases} 1, & \text{if } p^j \geq \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where b^j is the relevant label indicator for the jth class label, $\lambda \in (0.5, 1.0)$ is a purity threshold.

iii) We propose to use a label transfer mechanism to transfer the result of the relevant label vector to lower layers. Our idea is to preserve the identified relevant label

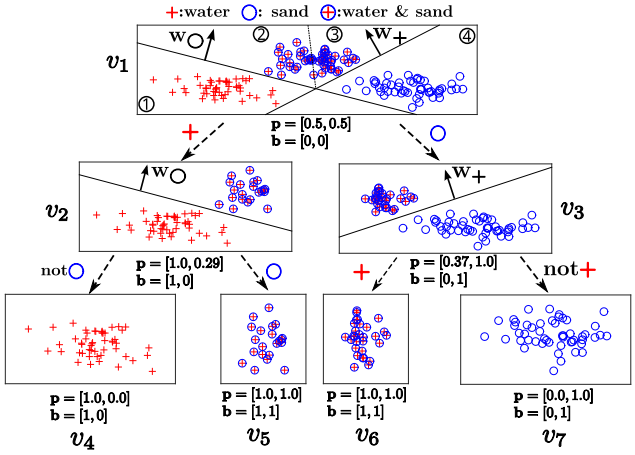

Fig. 2. An example of training procedure for multi-label classification.

vector $\mathbf{b}_p = [b_p^1, \ldots, b_p^q]^\top$ from the parent node and incorporate it as an additional indicator with the relevant label vector $\mathbf{b}_c = [b_c^1, \ldots, b_c^q]^\top$ of a child node to obtain a final result of relevant labels $\mathbf{b}$ as follows:

$$b^j = \begin{cases} 1, & \text{if } b_p^j = 1 \text{ or } b_c^j = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

iv) The above process continues until the stopping criterion is reached, i.e., the data cannot be further split by the induced classifiers.

Fig. 2 gives an example of two-label problem regarding Fig. 1 to illustrate the construction of ML-TREE. Note that each node in ML-TREE is with a set of one-against-all classifiers. Without lose of generality, we use linear SVM as the base classifier, and set the thresholding value $\lambda = 0.9$ as a default value. As shown in Fig. 2, the root node v_1 contains all training instances, and we train two one-against-all classifiers (i.e., $\mathbf{w}_0$ and $\mathbf{w}_+$) for the "0" and "+" classes, respectively (see Line 4 in Algorithm 2). We then classify the instance $\mathbf{x}$ according to the confidence scores, i.e., the margin values $(\mathbf{w}_0 \cdot \mathbf{x})$ and $(\mathbf{w}_+ \cdot \mathbf{x})$ (see Line 6 in Algorithm 2). As illustrated in node v_1, the solid lines are the resulting decision boundaries given by the binary classifiers, and the dash line is the combined boundary decided by the relative magnitudes of margins. Specifically, when $(\mathbf{w}_0 \cdot \mathbf{x}) > (\mathbf{w}_+ \cdot \mathbf{x}) > 0$, which is the case in region ③, $\mathbf{x}$ will be classified as "0" class. While if $(\mathbf{w}_+ \cdot \mathbf{x}) > (\mathbf{w}_0 \cdot \mathbf{x}) > 0$, which is the case in region ②, $\mathbf{x}$ will be classified as "+" class.

According to the decision surface, the instances in regions ① and ② would be classified as "+" class; while the instances in regions ③ and ④ would be classified as "0" class. Next, we use the prediction function to compute a vector of label probabilities $\mathbf{p}$ and a vector of relevant labels $\mathbf{b}$ w.r.t. each child node (see Lines 1 and 2 in Algorithm 2). For v_2, we have $\mathbf{p} = [1.0, 0.29]^\top$ and $\mathbf{b} = [1, 0]^\top$. According to the prediction criterion, the "+" class is considered as the relevant label for the instances in v_2. When building the classifier models for v_2, we do not need to consider the "+" class any more. In other words, we just need to train a classifier w.r.t. the "0" class for further splitting. The identified relevant label "+" will be transferred into the following child nodes, i.e., nodes v_4 and v_5. The construction process is similar for nodes v_3, v_6 and v_7 in the right sub-tree.

3.2 The ML-FOREST Algorithm

To improve the prediction performance, we further propose a ML-FOREST algorithm which extends the tree model using an ensemble method. A single standalone tree model can be assumed to partition the whole data space into regions belonging to different classes. However, it is likely that one single tree may overfit the data in the local region. In particular, it may make the inference of a test example unreliable. By applying an ensemble of trees, we first partition the whole data set into multiple random data subsets, and then construct multiple trees for each subset. In this way, we can greatly reduce the risk of overfitting on training data, and thus generally increase the overall prediction performance. More importantly, such strategy can greatly improve the scalability of the method over large-scale data sets. Lastly, its complexity is linear w.r.t. the number of trees. The ML-FOREST algorithm which builds an ensemble of K classifier trees $\{T_1, \ldots, T_K\}$ for multi-label classification is described in Algorithm 3,

Algorithm 3. ML-FOREST

Training Phase
Input: A training data set $\mathcal{D}$, the number of trees K
Output: A forest of tree classifiers $\mathcal{F}$
1: $\mathcal{F} = \phi$
2: **for** $i = 1$ to K **do**
3: prepare the training set $\mathcal{D}_i$ = bootstrap($\mathcal{D}$)
4: build tree classifier T_i = ML-TREE($\mathcal{D}$, none)
5: $\mathcal{F} = \mathcal{F} \cup T_i$
6: **end for**
7: **return** $\mathcal{F}$
Classification Phase
1: For a given $\mathbf{x}$, let $\mathbf{b}_1, \ldots, \mathbf{b}_K$ be the predictions assigned by the classifiers, calculate the confidence for each class, c^j, by the average combination method:
$$c^j = \frac{1}{K} \sum_{k=1}^{K} b_k^j$$
2: Assign $\mathbf{x}$ to the classes with the confidences larger than a predefined threshold value

An ensemble learner with higher diversity in component learners has better generalization performance. The theoretical and practical studies of ensemble diversity are well documented [40], [41]. In order to achieve diversity, we employ two randomization procedures to generate multiple hierarchical trees in ML-FOREST. First, each tree is trained on a data subset randomly drawn from the entire training set $\mathcal{D}$ using *sampled with replacement* [42]. In addition, the purity thresholding value λ for each tree is selected randomly in the range $(0.5, 1.0)$. Such a randomization procedure also frees us from fine-tuning an optimal λ value.

For the task of prediction, ML-FOREST outputs a confidence vector $\mathbf{c} = [c^1, \ldots, c^q]^\top \in \mathbb{R}^q$ for a testing example $\mathbf{x}$, where c^j represents the confidence for the jth class. To this end, we compute the predictions of all the trees regarding $\mathbf{x}$. For each tree, we seek a decision path from the root down to a leaf node based on the prediction of classifier at each

node. Based on the relevance label vectors (i.e., $\mathbf{b}_1, \ldots, \mathbf{b}_K$) from the leaves w.r.t. all K trees, we compute the ensemble confidence outputs $\mathbf{c}$ by

$$c^j = \frac{1}{K}\sum_{k=1}^{K} b_k^j, \tag{5}$$

where b_k^j is the jth element of the relevant label vector $\mathbf{b}_k$. More details about the prediction are given below.

3.3 Prediction Via Thresholding Strategies

For a testing example $\mathbf{x}$, ML-FOREST outputs a prediction vector $\mathbf{y} = [y^1, \ldots, y^q]^\top$ with $y^j = 1$ indicating the jth label is relevant regarding $\mathbf{x}$. Consider a confidence vector $\mathbf{c} = [c^1, \ldots, c^q]^\top \in \mathbb{R}^q$ for $\mathbf{x}$, where each element of $\mathbf{c}$ corresponds to a confidence value for one class label. Given $\mathbf{w}$, the prediction $\mathbf{y}$ of $\mathbf{x}$ can be completed by finding a bipartition of relevant and irrelevant labels based on a threshold function such that

$$y^j = \begin{cases} 1, & \text{if } w^j \geq t, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where $t \in [0, 1]$ is a predefined thresholding value. There are several ways to set the threshold value t. For example, we set $t = 0.5$ for simplicity.

Besides the above strategy, we can apply a max-drop thresholding scheme, called *Maximum Cut (MCut)* method [43], to find more flexible thresholding values for different examples data sets automatically. In this scheme, a testing example is first assigned with a set of relevant labels. Given a confidence output $\mathbf{c}$ for $\mathbf{x}$, we first sort the labels according to the values of $\mathbf{c}$, and then find two adjacent classes with the largest gap/difference in terms of their confidences. Lastly, we use the mean value as the threshold value for prediction.

3.4 Computational Complexity

Given a training data set containing m instances the proposed ML-TREE algorithm uses the one-against-all paradigm to build multi-class classifiers to partition the data at each node, the complexity of this node depends on the number of data instances $|\mathcal{D}_v|$ and the cost to train the classifiers.

To simplify the analysis, we assume that the number of learning data in the hierarchical tree is $m = N^d$, and the tree is a complete N-ary tree with d levels, which means the average branch-out is N and all leaf nodes are at the same level [22]. Let $f(|\mathcal{D}_v|)$ be the complexity of training one binary classifier at node v. The state-of-the-art algorithms for training a linear SVM classifier have a time complexity scaling close to $O(m)$ [44]. At the root node, we have a cost of Nm, while at the second level we have N additional cost of $N(\frac{m}{N})$, i.e., an additional cost of m. At the subsequent level, we have N^2 additional cost of $N(\frac{m}{N^2})$, and so on. Therefore, the total cost of the tree is $Nm(1 + \frac{1}{N} + \frac{1}{N^2} + \cdots + \frac{1}{N^d})$, which leads to $O(Nm)$ as a sum of a geometric series when d approaches infinity.

For the ensemble classifier with K trees, let the number of learning data in each tree be $m' = 0.632\, m$, and the average branch-out is N'. Thus the cost of every tree is $O(N'm')$. In addition, for each tree, we have an additional cost of $O(m)$ for the sampling. Consequently, the total complexity of the ensemble classifier is $K(O(N'm') + O(m))$.

Besides the complexity information of our approach, we also briefly analyze the complexities of several other well-known multi-label algorithms, i.e., binary relevance [10], classifier chain [38], and hierarchy of multi-label classifiers [22]. Similar to the analysis for ML-FOREST, let the time complexity for training a base classifier be $O(m)$ w.r.t. m training instances. BR decomposes a multi-label classification problem into q independent binary classification problems, thus the overall complexity is $O(qm)$. CC algorithm successively trains q binary classifiers, and appends the label focused by the last classifier as a new attribute to train the next classifier. By ignoring the extended label attribute, the complexity of CC is also $O(qm)$. For HOMER, as each training instance may pass through multiple paths from the root to leaves, it is difficult to analyze the complexity w.r.t. the number of training instances. In [22], it is shown that the complexity of HOMER is $O(f(q) + q)$, where q is the number of labels and $f(q)$ is the cost of the balanced clustering process used in HOMER. The running time comparison of these algorithms is provided in the experiment section.

4 EXPERIMENTS

4.1 Experimental Results on Synthetic Data Sets

In this experiment, we first use two synthetic data sets to validate whether our proposed ML-FOREST algorithm can mine reasonable label correlation or not. In the proposed ML-FOREST algorithm, we use the amount of reused label co-occurrences in the leaf nodes of the tree ensemble to automatically estimate the label correlation. Specifically, we construct a m-by-q matrix $\mathbf{Q}$ where m is the number of leaf nodes in the ensemble and q is the number of possible labels. The ith row of $\mathbf{Q}$ is equivalent to the relevant label vector $\mathbf{b}$ of the ith leaf node, where the entries with value 1 correspond to the non-zero elements of $\mathbf{b}$. The ith column of $\mathbf{Q}$ represents the distribution of label l_i over the leaves in the ensemble. The label correlation of two labels l_i and l_j is measured using the ϕ-coefficient defined as follows:

$$\phi(i, j) = (AD - BC)/\sqrt{(A + B)(C + D)(A + C)(B + D)}, \tag{7}$$

where A, B, C and D are the frequency counts of $l_i \wedge l_j$, $l_i \wedge \neg l_j$, $\neg l_i \wedge l_j$ and $\neg l_i \wedge \neg l_j$, respectively.

As we have no ground-truth label correlation for real-world data, we study two synthetic data sets [11] in which we know the exact label relationship. In the experiments, we use them to check the validity of our proposed method for capturing the label correlation. The data sets have 5,000 instances and five labels from l_1 to l_5. l_5 is assigned to an instance if it does not belong to l_1 to l_4. In the first data, we have $l_1 = l_2$ and $l_3 = l_4$. In the second data, we have $l_1 = l_2 \vee l_3 \vee l_4$. The label correlation information for these two data sets are given in Tables 1 and 2, respectively. The last row (column) of the tables for l_5 has negative references because l_5 is assigned to an instance if it belongs to none of l_1 to l_4. Table 1 shows that the entries $(1, 2)$, $(2, 1)$, $(3, 4)$ and $(4, 3)$ are 1.0, while Table 2 shows that the entries $(1, 2)$, $(1, 3)$,

TABLE 1
Label Correlation of the First Synthetic Data Set, $l_1 = l_2$ and $l_3 = l_4$

labels	l_1	l_2	l_3	l_4	l_5
l_1	1.00	1.00	-0.14	-0.14	-0.09
l_2	1.00	1.00	-0.14	-0.14	-0.09
l_3	-0.14	-0.14	1.00	1.00	-0.14
l_4	-0.14	-0.14	1.00	1.00	-0.14
l_5	-0.09	-0.09	-0.14	-0.14	1.00

TABLE 2
Label Correlation of the Second Synthetic Data Set, $l_1 = l_2 \vee l_3 \vee l_4$

labels	l_1	l_2	l_3	l_4	l_5
l_1	1.00	0.29	0.35	0.36	-0.03
l_2	0.29	1.00	0.30	0.23	-0.06
l_3	0.35	0.30	1.00	0.15	-0.03
l_4	0.36	0.23	0.15	1.00	-0.03
l_5	-0.03	-0.06	-0.03	-0.03	1.00

$(1, 4)$ have relatively large positive values. These results are consistent with the ground-truth label correlation.

4.2 Experimental Results on Real Data Sets

In this experiment, we compare the proposed ML-FOREST algorithm with 8 well-known multi-label classification algorithms on 12 benchmark multi-label data sets. For the purpose of reproducibility, we provide the code at: https://sites.google.com/site/qysite/.

4.2.1 Data Sets

Twelve multi-label data sets are used in the experiments, and these data sets are benchmark data sets from different application domains: *scene*, *emotions* and *corel5k* are image data sets, *genebase* and *yeast* are biology data sets, and the remaining seven are document corpus. Reuters(10), Reuters(21), and Reuters(90) are the Reuters-21578 text data sets w.r.t. the largest 10 classes, 21 classes, and 90 classes, respectively. All the data sets are originally split into training and test set [45]. The characteristics of the data sets are summarized in Table 3.

4.2.2 Parameter Instantiation

We compare the proposed Algorithm with eight well-known multi-label classification algorithms, i.e., binary relevance [10], classifier chain (CC) [38], multi-label k nearest neighbor (ML-kNN) [18], instance-based and logistic regression (IBLR-ML) [21], hierarchy of multi-label classifiers [22] and predictive clustering trees [32], random forest of predictive clustering trees (RF-PCT) [35] and Two stage architecture [36]. The implementations of the BR, CC, ML-kNN, IBLR-ML, HOMER, RF-PCT and TSA algorithms are based on the MULAN library[1], and the implementation of the PCT algorithm is based on the CLUS system.[2]

For the algorithms using base classifiers (i.e., BR, CC, HOMER, TSA and ML-FOREST), SVM with linear kernel in LIBSVM library [46] is used as the base classifier. The options "-b 1" is used to learn SVMs with probability outputs in the experiments. We use five-fold cross validation on the training set to select parameters. In particular, for each algorithm, the parameters yield the best average hamming loss using cross validation are selected. After that, the algorithm is trained again with the selected parameters on the whole training set and evaluated on the test set for the comparison. The ranges of parameters of different algorithms are given as follows. The number of neighbors k in ML-kNN and IBLR-ML is tuned from 5 to 30 with an increment of 5. The number of clusters in the HOMER is tuned within the range of 2 to 6. The values for parameter C of SVM for BR, CC, TSA and HOMER are tuned with the values $2^{-5}, 2^{-3}, \ldots, 2^{3}$. For RF-PCT, we set the number of models to 50. We try the feature subset sizes of $\log_2 f + 1$, $\sqrt{f}$, $f/10$ and $f/2$, where f is the total number of features. Such setting is similar to the testing in [47], and we find that $f/2$ results in yields the best performance, thus we use $f/2$ as the feature subset size in the model.

Our ML-FOREST algorithm has three essential parameters: the number of trees K, the purity threshold λ and the penalty parameter C of SVM base classifiers. For each tree, we randomly selected λ in the range $(0.9, 0.95)$. We set $K = 50$ and $C = 2^{-5}$ as default values. Unless otherwise stated, we use these default settings in the experiments.

4.2.3 Performance Measures

The performance of multi-label classification is measured by the *bipartition-based* metrics based on the comparison of the predicted labels of each example with the ground-truth labels provided by the data set. In our experiments, nine *bipartition-based* metrics (hamming loss, example-based accuracy, example-based precision, example-based recall, example-based F1, subset accuracy, macro-precision, macro-recall and macro-F1) are used to measure the performance. Please see [50] the detailed definitions of these metrics. The ranking-based metrics (e.g., one-error, coverage, ranking loss and average precision) compare the predicted ranking of the labels with the ground-truth ranking.

TABLE 3
Description of the Multi-Label Data Sets in Terms of the Number of Training (#tr.e.) and Test (#t.e.) Examples, the Number of Features (f), the Total Number of Labels(q), and the Label Cardinality (l_c)

data set	domain	#tr.e.	#t.e.	f	q	l_c
scene	Image	1,211	1,196	294	6	1.07
Reuters(10)	Text	6,490	2,545	500	10	1.11
Reuters(21)	Text	7,140	2,747	500	21	1.16
Reuters(90)	Text	7,770	3,019	500	90	1.24
medical	Text	333	645	1,449	45	1.25
genebase	Biology	463	199	1,186	27	1.25
ohsumed	Text	6,286	7,643	500	24	1.66
emotions	Image	391	202	72	6	1.87
tmc2007	Text	21,519	7,077	500	22	2.16
bibtex	Text	4,880	2,515	1,836	159	2.40
corel5k	Image	4,500	500	499	374	3.52
yeast	Biology	1,500	917	103	14	4.2

The data sets are ordered by their label cardinality (i.e., average number of labels per example).

1. http://mulan.sourceforge.net
2. http://clus.sourceforge.net.

TABLE 4
Performance of the Multi-Label Classification Algorithms in Terms of **Hamming Loss**↓

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.1028(5)	0.1062(7)	0.0960(2)	**0.0893(1)**	0.1180(8)	0.1332(9)	0.0977(4)	0.1041(6)	0.0966(3)
Reuters(10)	0.0165(3)	0.0166(4)	0.0218(8)	0.0208(7)	**0.0155(1)**	0.0288(9)	0.0187(6)	0.0160(2)	0.0183(5)
Reuters(21)	0.0110(2)	**0.0109(1)**	0.0141(8)	0.0130(7)	0.0115(4)	0.0214(9)	0.0129(6)	0.0111(3)	0.0128(5)
Reuters(90)	0.0050(2)	**0.0049(1)**	0.0054(6)	0.0058(8)	0.0053(5)	0.0118(9)	0.0057(7)	0.0051(3)	0.0052(4)
medical	**0.0119(1)**	0.0120(2)	0.0197(7)	0.0299(9)	0.0141(6)	0.0230(8)	0.0130(4)	0.0131(5)	0.0121(3)
genbase	0.0746(9)	0.0744(8)	0.0015(3)	0.0016(4)	0.0030(5)	0.0078(7)	**0.0007(1)**	0.0011(2)	0.0037(6)
ohsumed_norm	0.0434(2)	**0.0431(1)**	0.0564(5)	0.0569(6)	0.0480(4)	0.0624(9)	0.0595(8)	0.0580(7)	0.0450(3)
emotions	0.2137(3)	0.2459(4)	0.2830(9)	0.2797(8)	0.2112(2)	0.2591(6)	**0.1972(1)**	0.2632(7)	0.2564(5)
tmc2007	0.0553(3)	0.0554(4)	0.0611(8)	0.0583(6)	0.0578(5)	0.0759(9)	**0.0495(1)**	0.0550(2)	0.0601(7)
bibtex	0.0123(3)	0.0124(4)	0.0139(7)	0.0189(9)	0.0137(6)	0.0145(8)	0.0130(5)	0.0122(2)	**0.0120(1)**
corel5k	**0.0092(1)**	0.0093(2)	0.0094(3)	0.0231(9)	0.0103(7)	0.0096(5)	0.0095(4)	0.0119(8)	0.0100(6)
yeast	0.2048(4)	0.2196(8)	**0.1980(1)**	0.2005(3)	0.2119(7)	0.2426(9)	0.2065(6)	0.2060(5)	0.1985(2)
Avg. rank	**3.17**	3.83	5.58	6.42	5.00	8.08	4.42	4.33	4.17

We also consider other four ranking-based metrics (one-error, coverage, ranking loss, and average precision) for multi-label ranking evaluation. Ranking-based metrics are threshold independent. These measures compare the predicted ranking of the labels with the ground-truth ranking.

We use a two-step statistical test procedure (the corrected Friedman test and the post-hoc Nemenyi test) as recommended by Demšar [16] to compare the algorithms in a pairwise way across multiple data sets. The comparison results of the algorithms w.r.t. different evaluation metrics are given in Tables 4 to 17. The Friedman test is a non-parametric test for multiple hypotheses testing. The procedure involves ranking the algorithms (each row in each section of the table) in a descending order based on their performance for each data set separately in which the best performance for each data set gets the rank of 1.

The average rank of each algorithm across all the data sets (each column in each section of the table) is computed. Then, the Nemenyi post-hoc test is used in order to detect which algorithms are significantly different from each other, based upon the average ranks of the algorithms. The performances of two algorithms is significantly different if their average ranks differ by more than a critical distance (CD) whose value is depended on the number of algorithms, the number of data sets and a given significance level p. The results of the Nemenyi post-hoc test can be visually presented with diagrams [16] as shown in Fig. 3, where the critical distance is 3.468 and the significant level is $p = 0.05$. For each evaluation metric, the values on the horizontal axis show the average ranks of the algorithms in a manner that the algorithm has the best (worst) rank. The algorithms that do not differ significantly are connected by a bold horizontal line.

4.2.4 Results with Bipartition-Based Metrics

Fig. 3 shows the diagrams of the results on the multi-label classification data sets for the *bipartition-based* metrics. Tables 4 to 12 show the complete comparison results of the algorithms for different evaluation metrics. Each of the tables represents a specific evaluation metric. For each evaluation metric, the uparrow ↑ (downarrow ↓) indicates that a larger (smaller) value is more useful for such a specific evaluation metric. The numbers in brackets are the ranks of the algorithms in terms of one particular evaluation metric, and the numbers in boldface indicate the best ranking algorithms. The last row in each section of the table is the average ranks of the algorithms across all the data sets. According to the experimental results, ML-FOREST achieves competitive performance with the compared algorithms. The experiments also reveal a number of interesting points:

- The ML-FOREST, TSA, RF-PCT and HOMER methods are hierarchical methods using different strategies to exploit the label dependency. ML-FOREST,

TABLE 5
The Performance of the Multi-Label Classification Algorithms in Terms of **Accuracy** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.5811(7)	0.6702(2)	0.6240(5)	0.6495(4)	0.5838(6)	0.5698(9)	0.6608(3)	0.5811(8)	**0.7189(1)**
Reuters(10)	0.8945(6)	0.9038(3)	0.8600(9)	0.8706(7)	0.9054(2)	0.8602(8)	0.8991(5)	0.8994(4)	**0.9153(1)**
Reuters(21)	0.8689(4)	0.8796(2)	0.8553(6)	0.8511(8)	0.8731(3)	0.7794(9)	0.8537(7)	0.8684(5)	**0.8837(1)**
Reuters(90)	0.7741(6)	0.7874(3)	0.7788(5)	0.7860(4)	**0.7972(1)**	0.5678(9)	0.7295(8)	0.7738(7)	0.7932(2)
medical	0.7048(3)	0.7074(2)	0.3731(8)	0.4225(7)	0.6712(5)	0.2036(9)	0.6740(4)	0.6488(6)	**0.7590(1)**
genbase	0.3595(9)	0.3600(8)	0.9782(4)	0.9849(3)	0.9652(6)	0.9284(7)	**0.9916(1)**	0.9866(2)	0.9673(5)
ohsumed	0.5028(4)	0.5219(3)	0.3241(6)	0.2761(7)	0.5290(2)	0.1878(9)	0.2349(8)	0.3649(5)	**0.5571(1)**
emotions	0.5173(2)	0.5035(3)	0.3177(8)	0.3160(9)	0.4917(4)	0.4827(5)	**0.5804(1)**	0.4344(6)	0.4287(7)
tmc2007	0.5953(5)	0.5987(4)	0.5567(7)	0.5651(6)	0.6024(2)	0.4544(9)	**0.6393(1)**	0.5997(3)	0.5243(8)
bibtex	0.2865(5)	0.3067(4)	0.1294(8)	0.1651(7)	0.3352(2)	0.0462(9)	0.2299(6)	0.3179(3)	**0.3568(1)**
corel5k	0.0501(5)	0.0553(4)	0.0184(8)	0.0412(6)	0.1055(3)	0.0000(9)	0.0222(7)	**0.1934(1)**	0.1194(2)
yeast	0.4984(6)	0.4352(9)	0.4920(7)	0.5031(3)	0.5195(2)	0.4530(8)	**0.5341(1)**	0.4991(5)	0.5009(4)
Avg. rank	5.17	3.92	6.75	5.92	3.17	8.33	4.33	4.58	**2.83**

TABLE 6
The Performance of the Multi-Label Classification Algorithms in Terms of **Precision** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.6063(7)	0.7032(2)	0.6547(5)	0.6798(4)	0.6127(6)	0.5991(9)	0.6898(3)	0.6060(8)	**0.7573(1)**
Reuters(10)	0.9075(6)	0.9158(3)	0.8726(8)	0.8819(7)	0.9168(2)	0.8725(9)	0.9155(4)	0.9104(5)	**0.9409(1)**
Reuters(21)	0.8890(4)	0.8989(2)	0.8766(6)	0.8713(8)	0.8905(3)	0.8129(9)	0.8732(7)	0.8870(5)	**0.9340(1)**
Reuters(90)	0.8045(6)	0.8179(3)	0.8111(5)	0.8163(4)	0.8245(2)	0.5790(9)	0.7607(8)	0.8018(7)	**0.8524(1)**
medical	0.7450(3)	0.7470(2)	0.4163(8)	0.4668(7)	0.7183(5)	0.2574(9)	0.7276(4)	0.6875(6)	**0.8451(1)**
genbase	0.3613(9)	0.3617(8)	0.9899(6)	0.9975(2)	0.9807(7)	0.9950(3)	**1.0000(1)**	0.9950(4)	0.9941(5)
ohsumed	0.6373(4)	0.6609(2)	0.4339(6)	0.3708(7)	0.6473(3)	0.2656(9)	0.3387(8)	0.4646(5)	**0.7785(1)**
emotions	0.6229(4)	0.6007(5)	0.5107(8)	0.4942(9)	0.6345(3)	0.5817(6)	**0.6980(1)**	0.5489(7)	0.6619(2)
tmc2007	0.7607(4)	0.7627(3)	0.7243(8)	0.7283(7)	0.7337(6)	0.6481(9)	0.7699(2)	0.7603(5)	**0.7779(1)**
bibtex	0.4575(5)	0.4679(4)	0.2543(7)	0.2531(8)	0.4767(3)	0.1396(9)	0.3839(6)	0.4832(2)	**0.5656(1)**
corel5k	0.1298(4)	0.1157(5)	0.0425(8)	0.0680(7)	0.2146(3)	0.0000(9)	0.0770(6)	**0.3242(1)**	0.2680(2)
yeast	0.6994(4)	0.6966(5)	**0.7322(1)**	0.7069(3)	0.6686(7)	0.6370(9)	0.6678(8)	0.6941(6)	0.7171(2)
Avg. rank	5.00	3.67	6.33	6.08	4.17	8.25	4.83	5.08	**1.58**

TABLE 7
The Performance of the Multi-Label Classification Algorithms in Terms of **Recall** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.6217(8)	0.6877(3)	0.6488(5)	0.6576(4)	0.6346(6)	0.5711(9)	0.6948(2)	0.6325(7)	**0.7441(1)**
Reuters(10)	0.9111(6)	0.9183(5)	0.8750(8)	0.8849(7)	**0.9244(1)**	0.8714(9)	0.9200(4)	0.9213(3)	0.9227(2)
Reuters(21)	0.8897(5)	0.8993(2)	0.8743(6)	0.8668(8)	0.8968(3)	0.7816(9)	0.8710(7)	0.8933(4)	**0.9093(1)**
Reuters(90)	0.7942(7)	0.8056(2)	0.7963(5)	0.8027(4)	**0.8227(1)**	0.5683(9)	0.7383(8)	0.7963(6)	0.8051(3)
medical	0.7628(2)	0.7509(3)	0.3953(8)	0.5339(7)	0.7186(4)	0.2036(9)	0.7116(5)	0.6997(6)	**0.7822(1)**
genbase	0.9891(3)	0.9890(4)	0.9782(7)	0.9874(5)	0.9795(6)	0.9284(9)	**0.9916(1)**	0.9915(2)	0.9732(8)
ohsumed_norm	0.5479(4)	0.5679(3)	0.3579(6)	0.2961(7)	**0.6234(1)**	0.1878(9)	0.2417(8)	0.4484(5)	0.5681(2)
emotions	0.6089(2)	0.5965(4)	0.3639(8)	0.3622(9)	0.5371(6)	0.5842(5)	**0.7005(1)**	0.5998(3)	0.4893(7)
tmc2007	0.6940(4)	0.6930(5)	0.6542(7)	0.6546(6)	0.7383(2)	0.5322(9)	**0.7477(1)**	0.7036(3)	0.5334(8)
bibtex	0.2949(5)	0.3199(4)	0.1335(8)	0.2095(7)	0.3800(2)	0.0462(9)	0.2453(6)	0.3393(3)	**0.3952(1)**
corel5k	0.0515(6)	0.0587(5)	0.0190(8)	0.0772(4)	0.1225(3)	0.0000(9)	0.0223(7)	**0.2625(1)**	0.1423(2)
yeast	0.5822(5)	0.4950(9)	0.5491(8)	0.5783(6)	0.6327(2)	0.5860(4)	**0.6749(1)**	0.5903(3)	0.5730(7)
Avg. rank	4.75	4.08	7.00	6.17	**3.08**	8.25	4.25	3.83	3.58

TABLE 8
The Performance of the Multi-Label Classification Algorithms in Terms of **F1 Score** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.6030(8)	0.6870(2)	0.6426(5)	0.6623(4)	0.6102(6)	0.5800(9)	0.6819(3)	0.6062(7)	**0.7400(1)**
Reuters(10)	0.9045(6)	0.9128(3)	0.8694(8)	0.8792(7)	0.9156(2)	0.8682(9)	0.9115(4)	0.9103(5)	**0.9261(1)**
Reuters(21)	0.8826(5)	0.8927(2)	0.8691(6)	0.8631(8)	0.8870(3)	0.7913(9)	0.8661(7)	0.8828(4)	**0.9013(1)**
Reuters(90)	0.7907(6)	0.8035(3)	0.7949(5)	0.8012(4)	0.8148(2)	0.5712(9)	0.7421(8)	0.7902(7)	**0.8151(1)**
medical	0.7380(2)	0.7361(3)	0.3951(8)	0.4733(7)	0.7040(5)	0.2209(9)	0.7043(4)	0.6793(6)	**0.7945(1)**
genbase	0.5235(9)	0.5240(8)	0.9821(4)	0.9900(3)	0.9749(6)	0.9467(7)	**0.9941(1)**	0.9908(2)	0.9769(5)
ohsumed	0.5615(4)	0.5824(3)	0.3703(6)	0.3125(7)	0.6018(2)	0.2107(9)	0.2679(8)	0.4260(5)	**0.6265(1)**
emotions	0.5901(2)	0.5766(3)	0.3959(8)	0.3911(9)	0.5576(4)	0.5559(5)	**0.6675(1)**	0.5323(6)	0.5236(7)
tmc2007	0.6863(5)	0.6867(4)	0.6475(7)	0.6513(6)	0.6961(2)	0.5444(9)	**0.7265(1)**	0.6906(3)	0.6052(8)
bibtex	0.3343(5)	0.3559(4)	0.1619(8)	0.2087(7)	0.3944(2)	0.0671(9)	0.2764(6)	0.3718(3)	**0.4270(1)**
corel5k	0.0704(5)	0.0733(4)	0.0249(8)	0.0606(6)	0.1436(3)	0.0000(9)	0.0344(7)	**0.2774(1)**	0.1722(2)
yeast	0.6087(5)	0.5420(9)	0.5993(7)	0.6085(6)	0.6243(2)	0.5668(8)	**0.6459(1)**	0.6103(3)	0.6090(4)
Avg. rank	5.17	4.00	6.67	6.17	3.25	8.42	4.25	4.33	**2.75**

TSA and HOMER are based on problem transformation mechanism using SVM base classifiers to solve a hierarchy of partial binary classification problems, whilst RF-PCT is to utilize multiple component classifiers each deals with a partial data set. We observe that this type of methods consistently achieve better performance than the other methods, such as ML*k*NN and PCT are not competitive mainly due to the inadequacy of modeling label dependency. This suggests the importance of leveraging label dependency for multi-label classification.

- Comparing ML-FOREST with HOMER, we see that both ML-FOREST and HOMER have excellent overall performances. HOMER ranks 1st in terms of *recall* and *macro-recall*, while ML-FOREST is better than HOMER in terms of *precision*. Precision and recall are two different quantitative measures which evaluate the algorithm performance from different aspects. Precision is the fraction of predicted labels

TABLE 9
The Performance of the Multi-Label Classification Algorithms in Terms of **Subset Accuracy** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.5159(7)	0.6196(2)	0.5686(5)	0.6112(3)	0.5050(9)	0.5393(6)	0.5978(4)	0.5075(8)	**0.6564(1)**
Reuters(10)	0.8633(5)	0.8758(2)	0.8295(9)	0.8428(7)	0.8739(3)	0.8354(8)	0.8613(6)	0.8656(4)	**0.8833(1)**
Reuters(21)	0.8260(4)	**0.8391(1)**	0.8133(7)	0.8129(8)	0.8296(3)	0.7426(9)	0.8147(6)	0.8231(5)	0.8333(2)
Reuters(90)	0.7237(6)	0.7393(3)	0.7307(5)	0.7406(2)	**0.7436(1)**	0.5588(9)	0.6916(8)	0.7235(7)	0.7317(4)
medical	0.6078(3)	0.6248(2)	0.3070(7)	0.2853(8)	0.5736(5)	0.1535(9)	0.5829(4)	0.5581(6)	**0.6543(1)**
genbase	0.0000(8.5)	0.0000(8.5)	0.9648(3)	0.9648(4)	0.9296(6)	0.8844(7)	**0.9849(1)**	0.9749(2)	0.9397(5)
ohsumed_norm	0.3318(3)	0.3463(2)	0.1973(5)	0.1748(7)	0.3187(4)	0.1267(9)	0.1468(8)	0.1934(6)	**0.3718(1)**
emotions	0.2921(2)	0.2772(3)	0.1040(8)	0.1040(9)	0.2772(4)	0.2624(5)	**0.3119(1)**	0.1287(7)	0.1683(6)
tmc2007	0.3144(4)	0.3277(2)	0.2816(7)	0.2960(6)	0.3054(5)	0.2025(9)	**0.3534(1)**	0.3164(3)	0.2778(8)
bibtex	0.1678(5)	0.1817(3)	0.0549(8)	0.0708(7)	0.1825(2)	0.0036(9)	0.1125(6)	0.1765(4)	**0.2073(1)**
corel5k	0.0040(5)	0.0060(4)	0.0001(7)	0.0020(6)	**0.0180(1)**	0.0000(8.5)	0.0000(8.5)	0.0120(2)	0.0100(3)
yeast	0.1516(6)	0.1396(8)	0.1592(4)	0.1778(2)	**0.1963(1)**	0.1178(9)	0.1559(5)	0.1407(7)	0.1658(3)
Avg. rank	4.83	3.42	6.25	5.75	3.67	8.13	4.88	5.08	**3.00**

TABLE 10
The Performance of the Multi-Label Classification Algorithms in Terms of **Macro-Precision** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.7781(4)	0.7453(7)	0.7931(3)	**0.8314(1)**	0.6943(8)	0.6561(9)	0.7970(2)	0.7641(5)	0.7486(6)
Reuters(10)	**0.9123(1)**	0.9083(2)	0.8740(8)	0.8812(7)	0.9014(3)	0.8250(9)	0.8916(5)	0.9007(4)	0.8868(6)
Reuters(21)	0.8690(3)	0.8605(4)	0.8230(8)	0.8584(5)	0.8291(7)	0.6492(9)	**0.9202(1)**	0.8572(6)	0.8895(2)
Reuters(90)	0.5723(4)	**0.5952(1)**	0.5429(5)	0.4503(7)	0.5771(3)	0.0337(9)	0.4291(8)	0.5344(6)	0.5942(2)
medical	0.3372(4)	0.3571(3)	0.1477(8)	0.2023(7)	0.3130(5)	0.0185(9)	**0.3810(1)**	0.2665(6)	0.3665(2)
genbase	0.6994(7)	0.7041(4)	0.7037(5)	0.7391(2)	0.6139(8)	0.4059(9)	**0.8519(1)**	0.7325(3)	0.7037(6)
ohsumed	0.7206(2)	0.7077(3)	0.6331(5)	0.6524(4)	0.6288(6)	0.1189(9)	0.5935(8)	0.6016(7)	**0.7898(1)**
emotions	0.6842(3)	0.5956(7)	0.4751(9)	0.4773(8)	0.7051(2)	0.6194(4)	**0.7165(1)**	0.6006(6)	0.6043(5)
tmc2007	0.7812(5)	0.7664(6)	0.7376(7)	0.7828(4)	0.6951(8)	0.3948(9)	0.8458(2)	0.8042(3)	**0.9237(1)**
bibtex	0.5220(3)	0.5027(4)	0.1941(7)	0.1698(8)	0.4311(6)	0.0063(9)	0.4410(5)	0.5301(2)	**0.6249(1)**
corel5k	0.0516(5)	0.0523(4)	0.0334(7)	0.0329(8)	0.0553(3)	0.0000(9)	**0.3105(1)**	0.0616(2)	0.0516(6)
yeast	0.3729(6)	0.3231(9)	**0.6003(1)**	0.5101(3)	0.3460(8)	0.4099(4)	0.5626(2)	0.3677(7)	0.3932(5)
Avg. rank	3.92	4.50	6.08	5.33	5.58	8.17	**3.08**	4.75	3.58

TABLE 11
The Performance of the Multi-Label Classification Algorithms in Terms of **Macro-Recall** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.6134(8)	0.6770(3)	0.6374(5)	0.6442(4)	0.6270(6)	0.5605(9)	0.6792(2)	0.6250(7)	**0.7297(1)**
Reuters(10)	0.8237(4)	0.8368(3)	0.7539(8)	0.7415(9)	**0.8571(1)**	0.8062(5)	0.7886(6)	0.8487(2)	0.7769(7)
Reuters(21)	0.7460(4)	**0.7579(1)**	0.6980(5)	0.6952(6)	0.7564(3)	0.5442(9)	0.6171(8)	0.7577(2)	0.6875(7)
Reuters(90)	0.3983(4)	0.4014(3)	0.3551(7)	0.4058(2)	**0.4427(1)**	0.0375(9)	0.2161(8)	0.3772(6)	0.3809(5)
medical	0.3066(4)	0.3191(3)	0.0859(8)	0.2141(7)	0.2557(5)	0.0220(9)	**0.3498(1)**	0.2353(6)	0.3266(2)
genbase	0.7755(2)	0.7755(3)	0.6961(6)	0.7189(5)	0.6366(8)	0.4021(9)	**0.8519(1)**	0.7407(4)	0.6479(7)
ohsumed	0.4100(4)	0.4383(2)	0.2365(6)	0.2087(7)	**0.5308(1)**	0.0822(9)	0.1403(8)	0.3148(5)	0.4276(3)
emotions	0.6048(3)	0.5880(4)	0.3129(9)	0.3284(8)	0.5286(6)	0.5853(5)	**0.6824(1)**	0.6131(2)	0.4772(7)
tmc2007	0.4878(5)	0.5006(4)	0.4073(7)	0.4269(6)	0.5614(2)	0.2601(9)	0.3945(8)	0.5021(3)	**0.6275(1)**
bibtex	0.1698(5)	0.1922(4)	0.0508(8)	0.1177(7)	**0.2600(1)**	0.0063(9)	0.1366(6)	0.2098(3)	0.2372(2)
corel5k	0.0135(7)	0.0216(6)	0.0090(8)	0.0428(2)	0.0307(4)	0.0000(9)	**0.2995(1)**	0.0387(3)	0.0298(5)
yeast	0.3233(6)	0.2705(9)	0.3075(8)	0.3379(4)	0.3682(3)	0.3842(2)	**0.3980(1)**	0.3333(5)	0.3195(7)
Avg. rank	4.67	3.75	7.08	5.58	**3.42**	7.75	4.25	4.00	4.50

that are also relevant (evaluating the accurate of the prediction), while recall is the fraction of relevant labels that are predicted correctly (evaluating the completeness of the prediction). This result indicates that the predictions from ML-FOREST are more accurate than those from HOMER; while the predictions from HOMER are more complete than the ones from ML-FOREST. A reasonable explanation for this finding is that, HOMER uses multiple-leaf labeling method to classify a new multi-label example. HOMER returns the union of the predicted label set in multiple leaves, therefore the prediction is more complete. Whilst ML-FOREST is based on decision-path labeling method which combines the predictive labels from the root to a leaf as prediction result. The prediction at each node is obtained by the purity threshold λ with a large value in our experiments and therefore, is more accurate.

- Comparing ML-FOREST with RF-PCT, both ML-FOREST and RF-PCT consistently yield competitive, though

TABLE 12
The Performance of the Multi-Label Classification Algorithms in Terms of **Macro-F1** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.6853(7)	0.7063(4)	0.7003(5)	0.7226(3)	0.6575(8)	0.6025(9)	0.7322(2)	0.6865(6)	**0.7375(1)**
Reuters(10)	0.8635(4)	0.8692(3)	0.8050(8)	0.7998(9)	**0.8769(1)**	0.8132(7)	0.8338(5)	0.8720(2)	0.8239(6)
Reuters(21)	0.7996(3)	**0.8028(1)**	0.7408(7)	0.7612(6)	0.7876(4)	0.5576(9)	0.7112(8)	0.8005(2)	0.7780(5)
Reuters(90)	0.4503(3)	0.4574(2)	0.4077(7)	0.4090(6)	**0.4749(1)**	0.0336(9)	0.2630(8)	0.4256(5)	0.4403(4)
medical	0.3124(4)	0.3271(3)	0.1037(8)	0.1815(7)	0.2687(5)	0.0201(9)	**0.3581(1)**	0.2431(6)	0.3358(2)
genbase	0.7008(5)	0.7032(4)	0.6998(6)	0.7258(3)	0.6183(8)	0.4038(9)	**0.8519(1)**	0.7361(2)	0.6718(7)
ohsumed	0.5023(4)	0.5267(2)	0.3174(6)	0.3046(7)	**0.5437(1)**	0.0958(9)	0.1753(8)	0.3934(5)	0.5227(3)
emotions	0.6072(2)	0.5568(6)	0.3469(9)	0.3705(8)	0.5781(5)	0.5957(4)	**0.6800(1)**	0.6027(3)	0.5249(7)
tmc2007	0.5691(5)	0.5803(4)	0.4850(7)	0.5212(6)	0.6051(2)	0.2938(9)	0.4460(8)	0.5813(3)	**0.7009(1)**
bibtex	0.2241(5)	0.2470(4)	0.0692(8)	0.1308(7)	0.3007(2)	0.0063(9)	0.1834(6)	0.2678(3)	**0.3129(1)**
corel5k	0.0185(7)	0.0263(6)	0.0128(8)	0.0295(5)	0.0333(4)	0.0000(9)	**0.3008(1)**	0.0412(2)	0.0345(3)
yeast	0.3247(8)	0.2785(9)	0.3361(5)	0.3651(3)	0.3467(4)	**0.3914(1)**	0.3784(2)	0.3354(6)	0.3332(7)
Avg. rank	4.75	4.00	7.00	5.83	3.75	7.75	4.25	**3.75**	3.92

TABLE 13
The Performance of the Multi-Label Classification Algorithms in Terms of **One-Error** ↓

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.2425(3)	0.2776(7)	0.2533(6)	**0.2341(1)**	0.3294(8)	0.3813(9)	0.2475(4)	0.2366(2)	0.2475(5)
Reuters(10)	0.0530(2)	0.0534(4)	0.0896(8)	0.0864(7)	0.0727(6)	0.1218(9)	0.0625(5)	**0.0515(1)**	0.0531(3)
Reuters(21)	0.0641(2)	0.0648(3)	0.1012(8)	0.0910(6)	0.0961(7)	0.1744(9)	0.0823(5)	**0.0593(1)**	0.0673(4)
Reuters(90)	0.1447(3)	0.1414(2)	0.1534(5)	0.1613(6)	0.1650(8)	0.4193(9)	0.1630(7)	**0.1395(1)**	0.1467(4)
medical	0.1597(2)	0.1690(3)	0.3612(7)	0.5256(8)	0.2465(6)	0.6279(9)	0.2016(5)	0.1721(4)	**0.1504(1)**
genbase	**0.0000(2.5)**	0.0000(2.5)	0.0000(2.5)	0.0000(2.5)	0.0050(7)	0.0050(8)	0.0000(5)	0.0000(6)	0.0050(9)
ohsumed_norm	0.2281(3)	**0.2224(1)**	0.3892(6)	0.4117(7)	0.3202(4)	0.5393(9)	0.4438(8)	0.3598(5)	0.2239(2)
emotions	**0.2921(1)**	0.3614(7)	0.3615(8)	0.3564(6)	0.3515(5)	0.3812(9)	0.2970(2)	0.3020(3)	0.3020(4)
tmc2007	0.1710(4)	0.1749(5)	0.2040(7)	0.1946(6)	0.2374(8)	0.2984(9)	0.1542(2)	0.1600(3)	**0.1170(1)**
bibtex	0.3575(2)	0.3586(3)	0.6020(7)	0.6294(8)	0.4584(6)	0.7829(9)	0.4123(5)	0.3718(4)	**0.3458(1)**
corel5k	0.6760(3)	0.6860(4)	0.6940(5)	0.8820(9)	0.7580(7)	0.7680(8)	0.6400(2)	**0.5980(1)**	0.7140(6)
yeast	0.2443(7)	0.3162(9)	0.2345(2)	0.2410(4)	0.2497(8)	**0.2334(1)**	0.2443(6)	0.2356(3)	0.2410(5)
Avg. rank	**3.04**	4.21	5.96	5.88	6.67	8.17	4.67	2.83	3.75

TABLE 14
The Performance of the Multi-Label Classification Algorithms in Terms of **Coverage** ↓

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.5017(2)	0.5761(6)	0.5326(4)	0.5485(5)	1.1204(9)	0.9607(8)	0.5284(3)	**0.4900(1)**	0.6806(7)
Reuters(10)	0.2016(2)	0.2059(3)	0.3434(7)	0.3324(6)	0.5839(9)	0.3937(8)	0.2122(4)	**0.1737(1)**	0.3136(5)
Reuters(21)	0.3611(2)	0.3648(3)	0.7197(7)	0.5643(5)	1.6702(9)	0.9425(8)	0.3702(4)	**0.3189(1)**	0.6309(6)
Reuters(90)	2.0768(3)	2.1007(4)	2.9248(5)	3.1977(7)	12.7410(9)	7.1332(8)	1.7857(2)	**1.4495(1)**	3.0113(6)
medical	**2.1349(1)**	2.2977(3)	3.2775(6)	5.5488(7)	5.9922(9)	5.8977(8)	2.1457(2)	2.3752(4)	3.2000(5)
genbase	0.4925(3)	0.4925(4)	0.5678(8)	0.5025(5)	0.5226(6)	**0.3116(1)**	0.5477(7)	0.5779(9)	0.4774(2)
ohsumed	2.6219(2)	**2.4900(1)**	4.1917(4)	4.6049(7)	9.8917(9)	6.6931(8)	4.3671(5)	3.3690(3)	4.3869(6)
emotions	1.9455(2)	1.9851(4)	2.5149(8)	2.2970(6)	2.6634(9)	2.3515(7)	**1.9208(1)**	1.9505(3)	2.0545(5)
tmc2007	2.4972(6)	2.5796(7)	2.4054(5)	2.3213(4)	7.8205(9)	4.1345(8)	**1.8273(1)**	2.1683(2)	2.3071(3)
bibtex	23.5730(3)	24.2592(4)	61.0501(8)	48.7797(6)	74.1217(9)	58.5996(7)	18.6720(2)	**16.4091(1)**	25.4803(5)
corel5k	107.0980(3)	120.1340(6)	111.2760(4)	199.0800(8)	320.5720(9)	120.5880(7)	96.3340(2)	**92.1240(1)**	114.5120(5)
yeast	6.5638(7)	7.1145(9)	6.4144(2)	6.4264(3)	6.9586(8)	6.5354(5)	**6.2443(1)**	6.5431(6)	6.4885(4)
Avg. rank	3.00	4.50	5.67	5.75	8.67	6.92	2.83	**2.75**	4.92

perhaps not the best performance in terms of all the metrics. ML-FOREST and RF-PCT perform well in terms of *macro precision* and *macro recall* which take average of the precision and recall over different classes. This result implies that these two tree ensemble methods are robust across a range of different datasets and classes. Both of them are able to achieve good overall performance in the evaluation. The reason for this is that although growing each individual tree in ML-FOREST and RF-PCT may not be optimal, multiple fully grown trees can make it up for good and robust performance.

- We further analyze the performance of the ML-FOREST w.r.t. different types of data. We can see that the performances of ML-FOREST on text data sets (e.g., Reuters, ohsumed and bibtex) are better than those of the image and biology data sets. A reasonable explanation for this finding is that we use SVMs with a linear kernel as base classifier for ML-FOREST. Previous study [49] has shown that most text categorization

TABLE 15
The Performance of the Multi-Label Classification Algorithms in Terms of **Ranking Loss** ↓

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.0799(2)	0.0953(6)	0.0866(4)	0.0892(5)	0.2000(9)	0.1720(8)	0.0861(3)	**0.0780(1)**	0.1150(7)
Reuters(10)	0.0105(2)	0.0110(3)	0.0258(7)	0.0244(6)	0.0496(9)	0.0316(8)	0.0119(4)	**0.0084(1)**	0.0207(5)
Reuters(21)	0.0082(2)	0.0084(3)	0.0220(7)	0.0176(5)	0.0642(9)	0.0351(8)	0.0091(4)	**0.0069(1)**	0.0187(6)
Reuters(90)	0.0140(3)	0.0141(4)	0.0201(5)	0.0242(7)	0.1143(9)	0.0652(8)	0.0127(2)	**0.0095(1)**	0.0224(6)
medical	**0.0325(1)**	0.0351(3)	0.0540(6)	0.1029(7)	0.1030(8)	0.1135(9)	0.0344(2)	0.0366(4)	0.0507(5)
genbase	0.0045(4)	0.0046(5)	0.0060(7)	0.0037(2)	0.0059(6)	**0.0018(1)**	0.0061(8)	0.0077(9)	0.0043(3)
ohsumed_norm	0.0540(2)	**0.0507(1)**	0.1094(5)	0.1236(7)	0.2883(9)	0.2041(8)	0.1170(6)	0.0831(3)	0.1059(4)
emotions	0.1791(3)	0.1942(4)	0.2795(8)	0.2571(6)	0.3111(9)	0.2648(7)	**0.1711(1)**	0.1761(2)	0.2033(5)
tmc2007	0.0363(5)	0.0385(7)	0.0380(6)	0.0348(4)	0.1829(9)	0.0889(8)	**0.0208(1)**	0.0287(2)	0.0316(3)
bibtex	0.0781(3)	0.0792(4)	0.2399(7)	0.1961(6)	0.2972(9)	0.2556(8)	0.0620(2)	**0.0556(1)**	0.1085(5)
corel5k	0.1194(3)	0.1329(5)	0.1269(4)	0.2525(8)	0.5798(9)	0.1415(7)	0.1072(2)	**0.1009(1)**	0.1344(6)
yeast	0.1786(6)	0.2079(9)	0.1715(2)	0.1734(3)	0.2040(8)	0.1844(7)	**0.1706(1)**	0.1759(5)	0.1756(4)
Avg. rank	3.00	4.50	5.67	5.50	8.58	7.25	3.00	**2.58**	4.92

TABLE 16
The Performance of the Multi-Label Classification Algorithms in Terms of **Average Precision** ↑

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.8561(3)	0.8346(7)	0.8492(5)	0.8563(2)	0.7626(8)	0.7548(9)	0.8525(4)	**0.8591(1)**	0.8354(6)
Reuters(10)	0.9680(2)	0.9675(3)	0.9412(7)	0.9441(6)	0.9365(8)	0.9230(9)	0.9630(4)	**0.9711(1)**	0.9578(5)
Reuters(21)	0.9576(2)	0.9569(3)	0.9274(7)	0.9358(6)	0.9076(8)	0.8716(9)	0.9472(4)	**0.9612(1)**	0.9447(5)
Reuters(90)	0.8899(3)	0.8912(2)	0.8770(6)	0.8748(7)	0.8284(8)	0.6579(9)	0.8783(4)	**0.8989(1)**	0.8780(5)
medical	**0.8719(1)**	0.8646(2)	0.7289(7)	0.6049(8)	0.7581(6)	0.4915(9)	0.8393(5)	0.8572(3)	0.8438(4)
genbase	**0.9945(1)**	0.9943(2)	0.9939(3)	0.9924(6)	0.9864(9)	0.9912(8)	0.9927(5)	0.9924(7)	0.9938(4)
ohsumed_norm	0.8052(2)	**0.8115(1)**	0.6667(5)	0.6421(6)	0.6258(7)	0.4998(9)	0.6238(8)	0.7097(4)	0.7419(3)
emotions	0.7857(2)	0.7660(5)	0.7088(9)	0.7314(6)	0.7092(8)	0.7208(7)	**0.7942(1)**	0.7832(3)	0.7688(4)
tmc2007	0.8498(4)	0.8451(5)	0.8277(7)	0.8380(6)	0.7244(8)	0.7215(9)	0.8782(2)	0.8642(3)	**0.8793(1)**
bibtex	0.5699(3)	0.5697(4)	0.3291(8)	0.3349(7)	0.4151(6)	0.2118(9)	0.5612(5)	0.5925(2)	**0.6072(1)**
corel5k	0.2890(3)	0.2762(5)	0.2770(4)	0.1624(8)	0.1479(9)	0.2163(7)	0.3089(2)	**0.3471(1)**	0.2647(6)
yeast	0.7526(5)	0.7022(9)	**0.7585(1)**	0.7570(2)	0.7344(8)	0.7488(7)	0.7544(3)	0.7524(6)	0.7542(4)
Avg. rank	**2.58**	4.00	5.75	5.83	7.75	8.42	3.92	2.75	4.00

TABLE 17
The Performance of the Multi-Label Classification Algorithms in Terms of **Training Time** (in Hours) ↓

	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	RF-PCT	TSA	ML-FOREST
scene	0.5468(6)	0.8686(7)	0.0313(4)	0.0175(3)	3.1079(9)	**0.0004(1)**	0.0106(2)	0.0402(5)	2.4658(8)
Reuters(10)	0.3058(6)	0.4330(7)	0.1532(4)	0.1890(5)	3.5188(9)	**0.0019(1)**	0.0333(2)	0.0357(3)	1.6901(8)
Reuters(21)	0.6063(6)	0.7417(7)	0.1867(5)	0.1740(4)	7.5049(9)	**0.0049(1)**	0.0857(3)	0.0551(2)	2.9268(8)
Reuters(90)	1.0601(6)	1.1896(7)	0.0830(4)	0.6115(5)	16.9018(9)	**0.0173(1)**	0.0813(3)	0.0447(2)	2.0974(8)
medical	0.0092(5)	0.0098(6)	0.0010(2)	0.0054(4)	0.1623(8)	**0.0004(1)**	0.0016(3)	0.0619(7)	0.2783(9)
genbase	0.0943(6)	0.0912(5)	0.0214(4)	0.0011(3)	0.8938(7)	**0.0001(1)**	0.0006(2)	12.2302(9)	4.8198(8)
ohsumed	7.6682(5)	8.1229(6)	0.9197(3)	0.7229(2)	42.5799(8)	**0.2098(1)**	5.5266(4)	99.3969(9)	33.4865(7)
emotions	0.3317(7)	0.4120(8)	0.0009(3)	0.0015(4)	3.4787(9)	**0.0001(1)**	0.0004(2)	0.2355(6)	0.1796(5)
tmc2007	1.3500(3)	1.4708(4)	2.6801(7)	1.8375(5)	13.5168(8)	**0.0139(1)**	0.1480(2)	2.4727(6)	166.0751(9)
bibtex	22.6319(6)	29.4215(7)	0.4625(2)	2.5392(5)	39.7952(8)	**0.0541(1)**	0.9940(3)	1.7673(4)	59.8075(9)
corel5k	6.7812(5)	8.9521(6)	0.1062(2)	11.1200(8)	39.4257(9)	**0.0715(1)**	0.7692(4)	0.4930(3)	9.7242(7)
yeast	7.1219(7)	7.1248(8)	0.0266(5)	0.0015(2)	0.0216(4)	**0.0005(1)**	0.0057(3)	0.0630(6)	7.3917(9)
Avg. rank	5.67	6.5	3.75	4.17	8.08	**1.00**	2.75	5.17	7.92

problems are linearly separable and so linear kernel is often applied for text categorization tasks, while a radial basis kernel is more appropriate for image and biology data. In practice, the prior knowledge of which kernel function should be used is difficult to obtain in advance.

4.2.5 Results with Ranking-Based Metrics

In this section, we present results w.r.t. ranking metrics. Fig. 4 shows the results on the multi-label classification data sets in terms of ranking-based metrics. Tables 13 to 16 show complete comparison results of the algorithms.

The best performing methods are TSA and BR, followed by RF-PCT and ML-Forest. RF-PCT is robust across both bipartition-based metrics and ranking-based metrics. On the other hand, even though the HOMER approach is able to produce good results in terms of the example-based measures, it performs poorly across all the ranking-based evaluation measures. A similar observation is also found in recent studies [50], where it is shown that even sophisticate

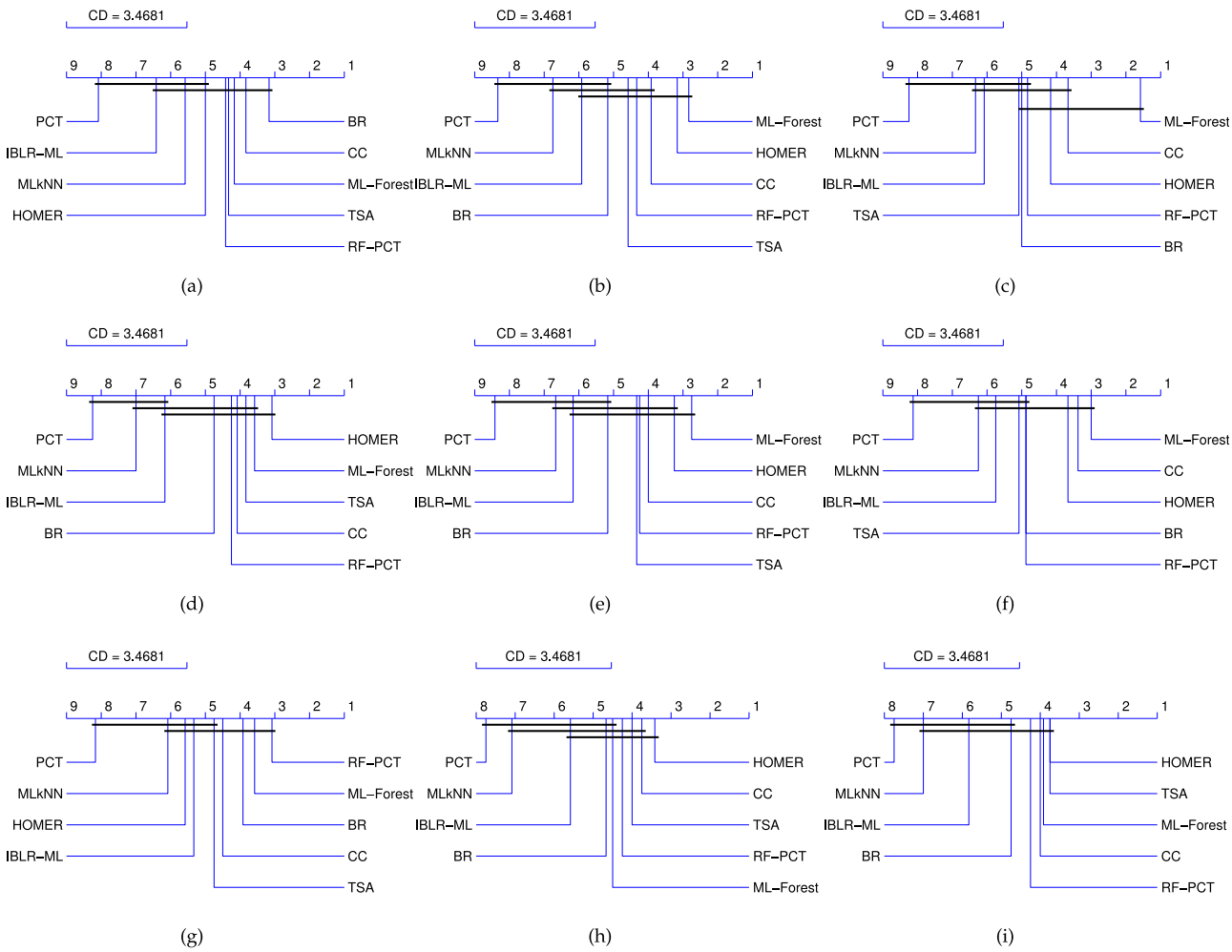

Fig. 3. The average ranks diagrams for the bipartition-based measures: (a) hamming loss, (b) accuracy, (c) precision, (d) recall, (e) F1-score, (f) subset accuracy, (g) macro-precision, (h) macro-recall, and (i) macro-F1.

approaches are not able to outperform all other methods in all measures. The evaluation measures used in the experiments assess the learning performance from different aspects and one algorithm rarely outperforms another algorithm on all criteria. With the results shown in Fig. 4 and Tables 13 to 16, we find that the performance of our proposed method is not significantly affected, it still can achieve competitive performance against TSA, BR, CC and RF-PCT in terms of the ranking-based metrics.

4.2.6 Parameter Sensitivity and Training Time

In this experiment, we investigate the influences of the number of trees K on the classification performance of ML-FOREST. We vary the values of K from 1 to 50. We try linear kernel, radial basis kernel, and polynomial kernel for SVM with their hyper-parameters selected using cross-validation. The accuracy results of ML-FOREST using different kernel functions against different number of trees K on the medical and scene data sets can be found in Table 18. When the number of trees increases, the accuracy of the ML-FOREST algorithm increases. The learning performance can be significantly enhanced when the ensemble has sufficient base learners. We can see from Table 18 that the performance of ML-FOREST does not have significant difference with different kernel functions.

We also evaluate the robustness of ML-FOREST against two parameters: the purity threshold λ and the SVM penalty C. The values 2^{-5}, 2^{-3}, ... 2 were considered for C and 0.4 to 0.9 for λ. The 3D graph in Fig. 5 shows how the accuracy of ML-FOREST varies against different values of λ and C. The accuracy of ML-FOREST increases when the value of λ increases and meanwhile the value of C decreases. We observe that the accuracy is degraded when λ is small. In this case, a number of irrelevant classes with confidences larger than λ are included within the predicted labels, and thus the performance degrades drastically. In this sense we randomly selected λ in the range $(0.9, 0.95)$ to set λ with a large value. The results on other data sets are similar. We also find that the performance of ML-FOREST will decrease if the penalty C is either too large or too small. In our experiment, we used the value $C = 2^{-5}$ as the default setting.

We also report the training times of each algorithm (see Table 17). The experiments are conducted on an Intel Xeon 2.4 GHz machine with 128 GB RAM running Windows Server 2012. The results show that PCT is the fastest method. The RF-PCT algorithm is implemented using multi-thread programming, and it ranks 2nd in terms of training time. Tree based learning methods, such as ML-FOREST and HOMER, have the longest training time. HOMER is

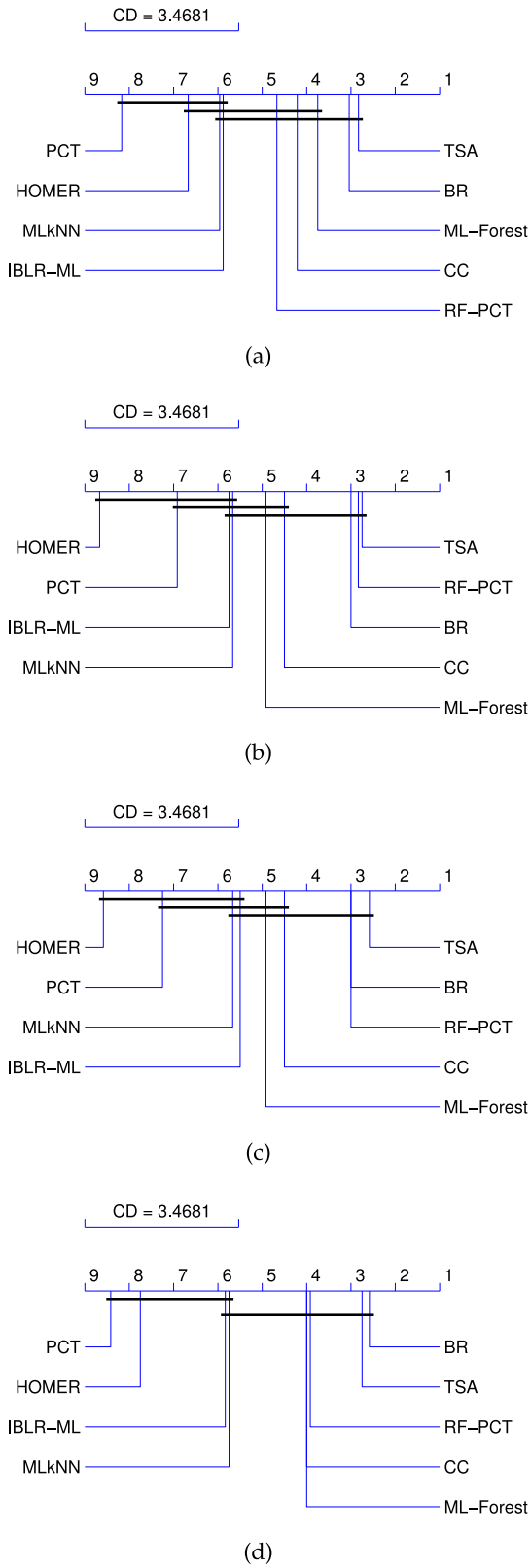

Fig. 4. The average ranks diagrams for the ranking-based measures: (a) one-error, (b) coverage, (c) ranking loss, and (d) average precision.

mainly due to the high computational cost in tuning the parameters of the model, while ML-FOREST is due to the high computational cost in learning multiple tree classifiers. The ML-FOREST algorithm is easy to be parallelized and thus the running time can be significantly improved in case using multi-thread parallel mechanism on computer with multi-core processors.

TABLE 18
The Accuracy of ML-FOREST Using Different Kernel Functions Against Different Number of Trees K on the Medical and Scene Data Sets

Data set	Kernel	1	5	10	20	50
medical	linear	0.696	0.728	0.734	0.743	0.751
	radial basis	0.719	0.719	0.728	0.737	0.742
	polynominal	0.729	0.732	0.738	0.744	0.747
scene	linear	0.608	0.705	0.717	0.718	0.719
	radial basis	0.585	0.714	0.718	0.718	0.718
	polynominal	0.609	0.612	0.710	0.715	0.719

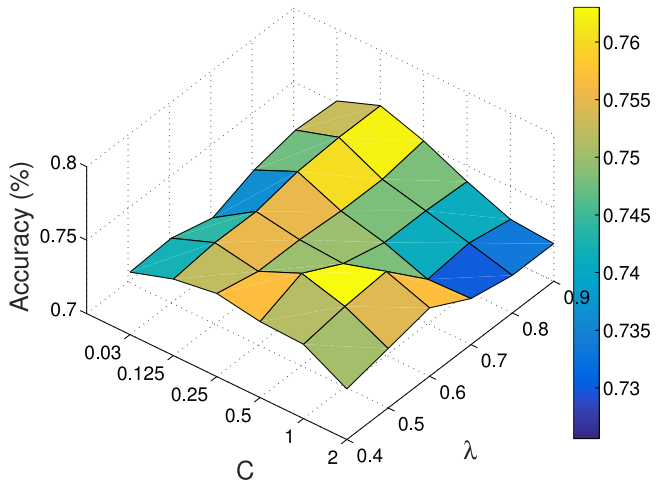

Fig. 5. The accurate of ML-FOREST with respects to the threshold λ and penalty C parameters.

5 CONCLUSION

In this paper, we have presented a new multi-label classification method, called ML-FOREST, to build an ensemble classifier. In ML-FOREST, we construct a set of hierarchical trees that are able to automatically exploit the label correlation, and develop a label transfer mechanism which identifies the relevant labels hierarchically.

ML-FOREST models the label dependency as a hierarchical scheme and performs the multi-label classification as a hierarchical decision process. As a result, ML-FOREST attains more discriminating ability than the first-order multi-label classification methods which simply transform a multi-label problem into multiple separate and independent binary problems. Experimental results show that the proposed tree ensemble method is highly competitive to the state-of-the-art multi-label classification algorithms. Several works remain to be investigated in our future work:

1) ML-FOREST is a hierarchical tree ensemble algorithm to model the label dependency. Instead of using the linear SVM as base classifier in the hierarchy, utilizing some probabilistic base classifier (such as Bayesian approach) might fit more to estimate the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$. This suggests one way to extend ML-FOREST.

2) Tsoumakas et al. [22] show that a hierarchical multi-label classifier model can be very efficient on the tasks with a large number of labels if clustering technique is considered to organize the labels in growing the tree. It is interesting to apply Tsoumakas's idea to ML-FOREST approach.
3) In practice, the acquired multi-label data set can be imbalanced and noisy, and thus the hierarchical tree can be imbalanced, which may degrade the generalization performance. Therefore, it is important to consider this challenging problem in the future.

ACKNOWLEDGMENTS

M. Tan is the corresponding author. The authors would like to thank the reviewers for their useful and constructive suggestions. They also thank Prof. Huaqing Min and Dr. Yuguang Yan for their contributions towards the revision of this work. This research was supported by the Guangzhou Key Laboratory of Robotics and Intelligent Software under Grant No. 15180007, the National Natural Science Foundation of China (NSFC) under Grant No. 61005061, the Guangdong Natural Science Foundation under Grant No. 2016A030313479, and the Fundamental Research Funds for the Central Universities under Grant Nos. D215048w and 2015ZZ029, and HKRGC CRF C1007-15GF and HKRGC GRF HKBU 12302715.

REFERENCES

[1] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers, "Statistical topic models for multi-label document classification," *Mach. Learn.*, vol. 88, no. 1/2, pp. 157–208, 2012.
[2] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification–revisiting neural networks," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2014, pp. 437–452.
[3] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, no. 1, pp. 2291–2320, 2012.
[4] K.-C. Chou, "Some remarks on predicting multi-label attributes in molecular biosystems," *Mol. Biosystems*, vol. 9, no. 6, pp. 1092–1100, 2013.
[5] X. Xiao, P. Wang, W.-Z. Lin, J.-H. Jia, and K.-C. Chou, "iAMP-2L: A two-level multi-label classifier for identifying antimicrobial peptides and their functional types," *Anal. Biochemistry*, vol. 436, no. 2, pp. 168–177, 2013.
[6] Y.-Y. Xu, F. Yang, Y. Zhang, and H.-B. Shen, "An image-based multi-label human protein subcellular localization predictor (ilocator) reveals protein mislocalizations in cancer tissues," *Bioinf.*, vol. 29, no. 16, pp. 2032–2040, 2013.
[7] R. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino, "Matrix completion for weakly-supervised multi-label image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 121–135, Jan. 2015.
[8] F. Sun, J. Tang, H. Li, G.-J. Qi, and T. S. Huang, "Multi-label image categorization with sparse factor representation," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 1028–1037, Mar. 2014.
[9] M. Liu, Y. Luo, D. Tao, C. Xu, and Y. Wen, "Low-rank multi-view learning in matrix completion for multi-label image classification," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2778–2784.
[10] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int. J. Data Warehousing & Min.*, vol. 3, no. 3, pp. 1–13, 2007.
[11] S. Huang, Y. Yu, and Z. Zhou, "Multi-label hypothesis reuse," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2012, pp. 525–533.
[12] K. Dembczynski, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence in multi-label classification," in *Proc. Workshop. Learn. Multi-Label Data*, 2010, pp. 5–12.
[13] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: Combining bayes with SVM," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 177–184.
[14] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2010, pp. 999–1008.
[15] L. Sun, S. Ji, and J. Ye, "Hypergraph spectral learning for multi-label classification," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2008, pp. 668–676.
[16] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
[17] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recog.*, vol. 37, no. 9, pp. 1757–1771, 2004.
[18] M. Zhang and Z. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recog.*, vol. 40, no. 7, pp. 2038–2048, 2007.
[19] G. Qi, X. Hua, Y. Rui, J. Tang, T. Mei, and H. Zhang, "Correlative multi-label video annotation," in *Proc. 15th Int. Conf. Multimedia*, 2007, pp. 17–26.
[20] S. Zhu, X. Ji, W. Xu, and Y. Gong, "Multi-labelled classification using maximum entropy method," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2005, pp. 274–281.
[21] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Mach. Learn.*, vol. 76, no. 2/3, pp. 211–225, 2009.
[22] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD'08 Workshop Min. Multidimensional Data*, 2008, pp. 30–44.
[23] 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, 2008.
[24] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2008, pp. 381–389.
[25] A. Clare and R. King, "Knowledge discovery in multi-label phenotype data," in *Proc. 5th Eur. Conf. Princ. Data Min. Knowl. Discovery*, 2001, pp. 42–53.
[26] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," *Adv. Neural Inf. Process. Syst.*, vol. 23, no. 1, pp. 163–171, 2010.
[27] J. Deng, S. Satheesh, A. C. Berg, and F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Proc. Adv. Neural Inf. Proc. Syst.*, 2011, pp. 567–575.
[28] G. Madjarov and D. Gjorgjevikj, "Hybrid decision tree architecture utilizing local SVMs for multi-label classification," in *Proc. 7th Int. Conf. Hybrid Artif. Intell. Syst.*, 2012, pp. 1–12.
[29] B. Fu, Z. Wang, R. Pan, G. Xu, and P. Dolog, "Learning tree structure of label dependency for multi-label learning," in *Proc. 16th Pacific-Asia Conf. Adv. Knowl. Discovery Data Min.*, 2012, pp. 159–170.
[30] W. Bi and J. T. Kwok, "Multi-label classification on tree-and dag-structured hierarchies," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 17–24.
[31] F. De Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *Proc. 3rd MLDM Int. Conf. Mach. Learn. Data Min. Pattern Recog.*, 2003, pp. 251–274.
[32] H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proc. 15th ICML Int. Conf. Mach. Learn.*, 1998, pp. 55–63.
[33] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma, "Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 13–24.
[34] J. Read, A. Puurula, and A. Bifet, "Multi-label classification with meta-labels," in *Proc. IEEE Int. Conf. Data Min.*, 2014, pp. 941–946.
[35] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Tree ensembles for predicting structured outputs," *Pattern Recog.*, vol. 46, no. 3, pp. 817–833, 2013.
[36] G. Madjarov, D. Gjorgjevikj, and S. Džeroski, "Two stage architecture for multi-label learning," *Pattern Recog.*, vol. 45, no. 3, pp. 1019–1034, 2012.
[37] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 279–286.

[38] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, 2011.
[39] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[40] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, 2003.
[41] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.
[42] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[43] C. Largeron, C. Moulin, and M. Géry, "MCut: A thresholding strategy for multi-label classification," in *Proc. 11th Int. Conf. Adv. Intell. Data Anal. XI*, 2012, pp. 172–183.
[44] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, 2005.
[45] G. Tsoumakas, J. Vilcek, L. Spyromitros, and I. Vlahavas, 'MULAN: A java library for multi-label learning," *J Mach Learn. Res.*, vol. 1, pp. 1–48, 2010.
[46] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst Tech.*, vol. 2, no. 3, 2011, Art. no. 27.
[47] D. F. Schwarz, I. R. König, and A. Ziegler, "On safari to random jungle: A fast implementation of random forests for high-dimensional data," *Bioinf.*, vol. 26, no. 14, pp. 1752–1758, 2010.
[48] Y. Ye, Q. Wu, J. Z. Huang, M. K. Ng, and X. Li, "Stratified sampling for feature subspace selection in random forests for high dimensional data," *Pattern Recog.*, vol. 46, no. 3, pp. 769–787, 2013.
[49] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. 10th Eur. Conf. Mach. Learn.*, 1998, pp. 137–142.
[50] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recog.*, vol. 45, no. 9, pp. 3084–3104, 2012.

Qingyao Wu received the BS degree in software engineering from the South China University of Technology, and the MS and PhD degrees in computer science from the Harbin Institute of Technology, China, in 2007, 2009, and 2013, respectively. He is currently an associate professor with the School of Software Engineering, South China University of Technology, China. He was a post-doctoral research fellow with the School of Computer Engineering, Nanyang Technological University, Singapore, from 2014 to 2015. His current research interests include machine learning, data mining, big data research, and deep learning.

Mingkui Tan received the bachelor's degree in environmental science and engineering and the master's degree in control science and engineering from Hunan University, Changsha, China, in 2006 and 2009, respectively, and the PhD degree in computer science from Nanyang Technological University, Singapore, in 2014. He is currently working as a senior research associate with the School of Computer Science, University of Adelaide, Australia. His research interests include compressive sensing, big data learning, and large-scale optimization.

Henjie Song is a professor with the School of Software Engineering, South China University of Technology, China. His research interests include artificial intelligence, machine learning, and data mining.

Jian Chen received the BS and PhD degrees in computer science from Sun Yat-Sen University, China, in 2000 and 2005, respectively. He is currently a professor with the School of Software Engineering (SSE), South China University of Technology, China. She joined SSE at the South China University of Technology as a faculty member in 2005. Her research interests include data mining, information retrieval, and recommendation techniques.

Michael K. Ng received the BSc and MPhil degrees from the University of Hong Kong, in 1990 and 1992, respectively, and the PhD degree from the Chinese University of Hong Kong in 1995. He is the head and chair professor of the Department of Mathematics, and professor (Affiliate) of the Department of Computer Science at the Hong Kong Baptist University. He was a research fellow of Computer Sciences Laboratory, Australian National University (1995-1997), and an assistant/associate professor (1997-2005) at the University of Hong Kong before joining Hong Kong Baptist University. His research interests include bioinformatics, data mining, image processing, scientific computing, and data mining, and he serves on the editorial boards of international journals. See http://www.math.hkbu.edu.hk/∼mng.