ML-TREE: A Tree-Structure-Based Approach to Multilabel Learning

Qingyao Wu, *Student Member, IEEE*, Yunming Ye, *Member, IEEE*, Haijun Zhang, Tommy W. S. Chow, *Senior Member, IEEE*, and Shen-Shyang Ho, *Member, IEEE*

***Abstract*— Multilabel learning aims to predict labels of unseen instances by learning from training samples that are associated with a set of known labels. In this paper, we propose to use a hierarchical tree model for multilabel learning, and to develop the ML-Tree algorithm for finding the tree structure. ML-Tree considers a tree as a hierarchy of data and constructs the tree using the induction of one-against-all SVM classifiers at each node to recursively partition the data into child nodes. For each node, we define a predictive label vector to represent the predictive label transmission in the tree model for multilabel prediction and automatic discovery of the label relationships. If two labels co-occur frequently as predictive labels at leaf nodes, these labels are supposed to be relevant. The amount of predictive label co-occurrence provides an estimation of the label relationships. We examine the ML-Tree method on 11 real data sets of different domains and compare it with six well-established multilabel learning algorithms. The performances of these approaches are evaluated by 16 commonly used measures. We also conduct Friedman and Nemenyi tests to assess the statistical significance of the differences in performance. Experimental results demonstrate the effectiveness of our method.**

***Index Terms*— Hierarchical tree model, multilabel classification, multilabel learning, tree-based classification.**

I. Introduction

EACH entity in nature may have multiple labels. For example, a patient may suffer from a few diseases; a gene may be associated with a number of functional classes; a document may cover several topics. This drives the need for effective approaches to cope with the multilabel learning problem. Because of the potential applications, such as text categorization [1]–[4], bioinformatics [5]–[7], image classification [8], [9], video annotation [10], and audio emotion detection [11], multilabel learning has caught broad attention in the research community over the past few years.

One simple approach to solve the multilabel problem is to decompose it into a set of single-label problems and to train independent binary classifier for each class label. This approach is known as the binary relevance (BR) method [12]. However, the main drawback of this approach lies in the fact that it ignores the correlations among labels. The label sets of different multilabel instances may overlap and the presence or absence of the labels for an instance needs to be predicted at the same time via exploiting correlations between these labels, which usually cannot be accomplished using BR. Therefore, it is desirable to develop more effective methods to explicitly capture the label correlation knowledge to increase prediction accuracy for multilabel learning.

The contribution of this paper is a new hierarchical tree approach, called ML-Tree, which encodes the label correlation information in a tree classification model for multilabel learning. The set of multilabel training data can be represented in the form of a tree structure where the root node corresponds to all the instances in the data, and the leaves correspond to the resulting fragments of the instances. For this approach, one-against-all SVM classifiers (with winner-takes-all strategy) are applied at each node to recursively partition the training instances into smaller subsets while moving down the tree. This process continues until the remaining instances at the node cannot be further split by the induced classifiers. For each node, we define a predictive label vector (PLV) to represent the predictive label transmission in the hierarchical tree model. The predictive labels are shared across the tree model and they are transferred in a top-down manner. Intuitively, if the instances at a node v are labeled with a class l, all instances of the child nodes of v will inherit the label l. In the training phase, PLV serves as a filter to select the individual one-against-all classifiers for learning at each node. In the testing phase, PLV ensures that a multilabel prediction at a leaf node is the integration of labeling results of all the nodes from the root down to the leaf node. With the propagation of the predictive labels, the correlations between labels can be preserved. Unlike the BR approach, our ML-Tree method provides a new multilabel classification framework, where the

Manuscript received May 25, 2013; revised September 6, 2013 and January 3, 2014; accepted March 28, 2014. Date of publication April 29, 2014; date of current version February 16, 2015. This research was supported in part by National Key Technology R&D Program of MOST China under Grant No. 2012BAK17B08, and NSFC under Grant No.61272538 and Grant 61300209, and Shenzhen Foundation Research Fund under Grant JCY20120613115205826, and Shenzhen Technology Innovation Program under Grant CXZZ20130319100919673, and Research Grants Council of the Hong Kong Special Administrative Region (CityU8/CRF/09), and AcRF Grant RG-41/12 and Singapore NTU Start-Up Grant.

Q. Wu, Y. Ye, and H. Zhang are with the Shenzhen Key Laboratory of Internet Information Collaboration, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: wuqingyao.china@gmail.com; yeyunming@hit.edu.cn; aarhzhang@gmail.com).

T. W. S. Chow is with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: eetchow@cityu.edu.hk).

S. S. Ho is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: ssho@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2315296

original multilabel classification is cast into the hierarchical nested one-against-all classification problems, and learning classifiers at different levels of a tree can effectively work together to reveal the correlations among labels. A multilabel classification can then be obtained by evaluating the node classifiers in a simple top-down fashion.

We examine the ML-Tree method on 11 real data sets from different domains and compare with six well-established multilabel learning algorithms in the experiments, that is, binary relevance (BR), classifier chain (CC), multilabel k nearest neighbor (ML-kNN), instance-based and logistic regression (IBLR-ML), hierarchy of multilabel classifiers (HOMER), and predictive clustering trees (PCT). We evaluate the performance of the algorithms by using 16 different measures. We also analyze the experimental results using Friedman and Nemenyi tests to assess the statistical significance of the performance differences. The experimental results highlight the effectiveness of our proposed ML-Tree method.

The remaining sections of the paper are organized as follows. In Section II, we describe the task of multilabel learning and give a brief overview of the related work. In Section III, a detailed description of our proposed method is presented. Extensive experimental results are given in Section IV. Conclusions are drawn and future work are discussed in Section V.

II. Background

In this section, we describe the task of multilabel learning and present an overview of the methods for multilabel learning. Tsoumakas and Katakis [12] summarize the multilabel learning methods into two groups: problem transformation methods and algorithm adaptation methods. It is worth noting that tree-based methods for multilabel learning are closely related to our proposed method. Therefore, in this section we also review a number of well-known tree-based methods for multilabel learning. An extensive review on multilabel learning is found in [14].

A. Task of Multilabel Learning

Let $\mathcal{X} = \mathcal{R}^d$ be the d-dimensional instance space and $\mathcal{Y} = \{1, 2, \ldots, q\}$ be the label space with a finite set of q possible labels. We can distinguish two types of multilabel learning tasks: multilabel classification and multilabel ranking. The task of multilabel classification is to learn a function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ from a given data set $\{(x_1, Y_1), (x_2, Y_2), \ldots, (x_m, Y_m)\}$, where $x_i \in \mathcal{X}$ is an instance, and $Y_i \subseteq \mathcal{Y}$ is a set of labels $\{Y_{i,1}, Y_{i,2}, \ldots, Y_{i,q}\} \in \{0, 1\}^q$. The task of multilabel ranking is to learn a scoring function $s : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ that assigns a real number indicating the relevance of an instance to a class label. The score $s(x, l)$ corresponds to the relevance of label l to instance x. The main product of the learning algorithms is a ranking of class labels $rank_s(x, l)$ ordered by the scores of the class labels.

B. Problem Transformation Methods

A single-label problem can be regarded as a degenerated problem of multilabel learning, thus it is natural to solve a multilabel problem by decomposing it into a set of single-label problems. Traditional supervised learning techniques can be applicable to the decomposed single-label problems. There are three types of problem transformation methods: binary relevance (BR), label powerset (LP) and pair-wise methods (PW). Suppose the number of labels of a multilabel problem is up to q. BR uses the one-against-all strategy to convert the multilabel problem into q binary classification problems [13]. However, it supposes all the binary classification problems are independent and does not consider label correlations. The LP method is to transform the multilabel problem into a single multiclass problem with 2^q labels using the power set of labels as the set of possible labels [13]. All possible label subsets from the original multilabel space are represented in the new single-label space. In this way, LP-based methods directly take the label correlations into account. Notwithstanding, the drawback is that the space of possible label subsets can be very large. The PW method learns $q \cdot (q-1)/2$ classifiers that cover all pairs of labels. Each classifier is learnt using the samples of the first label as positive examples and the samples of the second label as negative examples. All classifiers are then combined to make predictions by majority voting method. PW incurs the heavy computational cost since all possible pairwise classifiers are included.

C. Algorithm Adaptation Methods

The algorithm adaptation methods are able to handle multilabel data directly by extending and customizing existing machine learning algorithms. A number of algorithm adaptation methods for multilabel learning have been proposed, such as svm [5] kNN [9], neural networks [6], probabilistic generative models [4], maximum entropy methods [15], maximal margin methods [3], graphical models [16], and random decision tree [17]. One of the most popular methods is the BoosTexter algorithm proposed by Schapire and Singer [1]. It includes two boosting extensions: AdaBoost.MH and AdaBoost.MR. AdaBoost.MH is designed to minimize the Hamming loss, and Adaboost. MR is developed to find a hypothesis that places the correct labels at the top of the ranking list. Multilabel k-nearest neighbor (ML-kNN) is proposed by Zhang and Zhou [9]. It extends the lazy/instance-based learning algorithm, kNN, for multilabel learning, adopts the label prior probabilities obtained from each instance's k nearest neighbors, and then uses maximum-a-posterior (MAP) principle to determine the relevant labels. Recently, Cheng and Hüllermeier [18] unify the instance-based learning and the logistic regression (IBLR-ML) for multilabel classification. Their basic idea is to consider the information derived from instances similar to a query instance as the features of that instance; thereby, to some extent, blurring the distinction between instance-based and model-based learning. Zhang and Zhou [6] develop the BP-MLL algorithm which adapts the back-propagation neural network algorithm for multilabel learning. The main difference is that it considers multiple labels with the introduction of a new error function. Elisseeff and Weston [5] propose a ranking approach for

multilabel learning based on SVM algorithm where they use the average fraction of incorrectly ordered pairs of labels as cost function.

D. Tree-Based Methods

The methods that adapt the tree structure for the task of multilabel learning are referred as the tree-based methods. Clare and King [7] propose the ML-C4.5 algorithm which adapts the C4.5 algorithm for multilabel data by modifying the formula of entropy calculation. Comité *et al.* [2] extend an alternative decision tree to handle multilabel data, where the AdaBoost.MH algorithm proposed by Schapire and Singer [1] is employed to train the multilabel alternating decision trees. Blockeel *et al.* [19] propose the concept of predictive clustering trees (PCTs). Vens *et al.* [20] introduce several approaches for the PCTs algorithm to hierarchy multilabel classification where instances may belong to multiple classes and these classes are organized in a hierarchy. Kocev *et al.* [21] consider two ensemble learning techniques, bagging and random forests, and apply them to PCTs for multilabel learning. Tsoumakas *et al.* [23] propose the HOMER algorithm to handle data sets with a large number of labels. HOMER partitions the whole label set into disjointed subsets. The partition process is implemented by a balance clustering algorithm. Bengio *et al.* [24] put forward an algorithm for learning a tree structure of classifiers by optimizing the overall tree loss. This algorithm is originally proposed for multiclass problem, but its way of constructing a label tree is also applicable to multilabel problem. Punera *et al.* [25] develop a new technique that extracts a suitable hierarchical structure automatically from a corpus of label documents. Deng *et al.* [26] present a novel approach to learn a label tree for learning a large-scale classification with many classes efficiently. Madjarov and Gjorgjevikj [27] build decision trees for multilabel classification, where the leaves contain SVM-based classifiers to provide multilabel predictions. Fu *et al.* [28] construct a tree structure of the labels to describe the label dependency in multilabel data. Cesa-Bianchi *et al.* [29] study the hierarchical classification problem and introduce a refined evaluation scheme to turn the hierarchical SVM classifier into an approximated Bayes optimal classifier. They also study the problem of hierarchical classification in a taxonomy that allows classifications to be associated with multiple and/or partial paths. They further propose a new incremental algorithm to learn classifier for each node of the taxonomy [30]. Bi and Kwok [31] provide a novel hierarchical multilabel classification algorithm which can be used on both tree and DAG structure hierarchies. Zhang and Zhang [32] use a Bayesian network structure to encode the conditional dependencies of the labels and the feature set efficiently, with the feature set as the common parent of all labels. In this network, multilabel learning is decomposed into a series of single-label classification problems. Zhang and Zhang [32] describe the multilabel problem from the perspective of the Bayesian probability and classify the multilabel learning methods into the first-order, the second-order, and high-order approaches on the basis of the order of label correlations in a

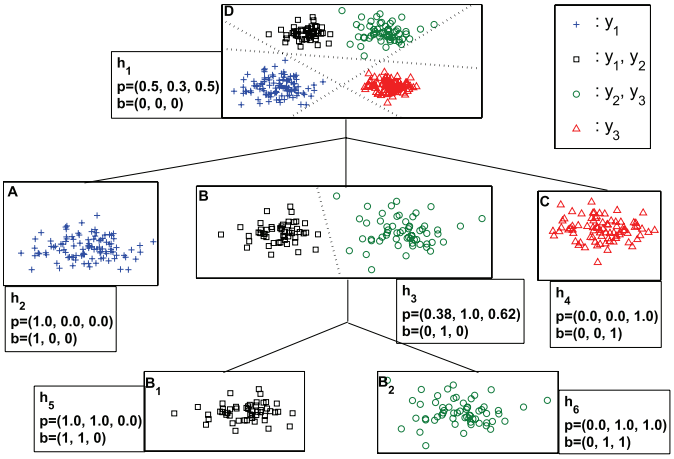

Fig. 1. Example to illustrate the process of building a hierarchical tree model. h_i: the one-against-all SVM classifiers learnt at each node. Dash lines: the decision boundaries of h_i. **b**: the predictive label vector indicating the predictive labels of a node. **p**: the class purity vector representing the purity of the classes of a node. In the example, we set $\lambda = 0.8$, meaning that the classes with purity value $p_v(l) \geq 0.8$ are predictive labels and their corresponding value of $b_v(l)$ are set to be 1. The predictive labels of a parent node will be transferred to its child nodes.

system. Our proposed method is different from these previous works in the way that the predictive labels are formulated at each node of the hierarchy. The predictive labels are shared across the hierarchical tree structure and transferred in a top-down manner. With this assumption, the original multilabel classification is built into the nested single-label prediction problems in a hierarchical way, which preserves the correlations between multiple labels.

III. ML-Tree for Multilabel Learning

A. Hierarchical Tree Model

In this paper, we develop a new hierarchical tree algorithm for multilabel learning, that is, ML-Tree. A hierarchical tree is constructed where the root node corresponds to all the instances in the training data set D, the instances are recursively partitioned into smaller subsets while moving down the tree. Each internal node v contains the union of the training instances of its child nodes, $D_v = \cup_{i=1}^{k} D_{c_i}$, $c_i \in \text{children}(v)$ and $D_{c_i} \cap D_{c_j} = \emptyset$, $i \neq j$ for $i, j = 1, 2, \ldots, k$. At each node v of the tree, one-against-all SVM classifiers are used to partition the corresponding data set D_v into disjoint subsets. This process continues until the remaining instances at the node cannot be further split by the induced classifier.

Fig. 1 shows an example of a hierarchical tree constructed from a multilabel data set containing three labels in 2-D feature spaces. Each node v contains a set of one-against-all SVM classifiers h_v. Dash lines at the node: the decision boundaries given by the classifiers. The top node contains all training data. At the top level, the whole data set is partitioned into three data subsets $\{A, B, C\}$. The second subset B is further partitioned into two subsets $\{B_1, B_2\}$ at the bottom level.

Each node in the tree contains two important components, that is, class purity vector and predictive label vector (see Fig. 1), for handling multilabel learning in training phase as well as prediction in testing phase. Suppose we have a set of multilabel training instances D_v at node v, $x_i \in D_v$

is an instance, and the class membership vector of x_i is $Y_i = \{Y_{i,1}, Y_{i,2}, \ldots, Y_{i,q}\} \in \{0, 1\}^q$.

Definition 1: The class purity vector (CPV), denoted as $\mathbf{p}$, is a vector with real values to measure the purity of the class labels at a node. The purity of a class l at a node is defined as the proportion of training instances belonging to class l at the node.

The CPV of node v is denoted as $\mathbf{p}_v$, and it is given by

$$p_v(l) = \frac{\sum_{x_i \in D_v} Y_{i,l}}{|D_v|} \tag{1}$$

where $p_v(l)$ is the lth component (for class l, $1 \leq l \leq q$) of $\mathbf{p}_v$ with values in the range of [0, 1], and $|D_v|$ is the total number of instances in D_v.

The class purity $p_v(l)$ can be used to estimate the prior probability that an instance at node v with the class l. If $p_v(l)$ is larger than a threshold, it implies that class l is the proper label for instances of node v. In this case, we call class (label) l a predictive label; otherwise, we call class (label) l a nonpredictive label with respect to node v. A nonpredictive label indicates that we cannot determine whether, at node v, the instances belong to class l or not. This decision is determined by the child nodes of v.

Definition 2: The PLV, denoted as $\mathbf{b}$, is a vector with boolean values indicating the membership of predictive labels at a node, where its lth component $b(l)$ ($1 \leq l \leq q$) takes the value of 1 if class l is a predictive label ($b(l) = 1$) and 0 otherwise ($b(l) = 0$).

The PLV of node v is denoted as $\mathbf{b}_v$, and each of its components is computed as follows:

$$b_v(l) = \begin{cases} 1, & \text{if } b_{v'}(l) = 1 \text{ or } p(l) \geq \lambda \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where λ ($0 \leq \lambda \leq 1$) is a threshold, and $v' = \text{parent}(v)$ is the parent node of node v. The predictive label of parent node v' is transferred to its child nodes, that is, $b_v(l) = 1$ in the case that $b_{v'}(l) = 1$.

PLV plays an important role in our ML-Tree method. In the training phase, predictive labels can be used as a filter to select the individual one-against-all SVM classifiers for learning at each node, that is, only each of the nonpredictive labels will be trained with a one-against-all SVM. In the testing phase, the multilabel classification for an unseen instance x is obtained according to the predictive labels of the leaf node that x goes through.

PLV has the following property. If the instances at a node v are labeled with a class l, all instances at each child node of v will inherit the label l. In our tree model, we incrementally learn PLVs along a decision path from the root of the tree down to a leaf node. As a result, a PLV at a leaf node is the combination of the predictive labels of all the nodes in the path. This approach enables us to model the multilabel classification problems in a hierarchical form, making it easy to exploit the correlations among labels in multilabel learning.

As shown in Fig. 1, each node is associated with a PLV. The leaf nodes B_1 and B_2 are labeled with y_1, y_2 and y_2, y_3, respectively. Their parent node B is labeled with class y_2.

TABLE I
PSEUDOCODE OF THE LEARNING ALGORITHM

ML-Tree(D)

Input: D, the current training instances.
Output: a hierarchical tree for multi-label learning.

1) create a node v;
2) compute vector $\mathbf{p}_v$ and $\mathbf{b}_v$ to identify the predictive labels for node v according to equations (1) and (2) respectively.
3) apply the induced h_v derived from the function **Splitting**(D, $\mathbf{b}_v$) to find a partition $\mathcal{P}$ on D;
4) **if Stopping**(D, $\mathbf{b}_v$) is TRUE then return v as a leaf node and attach h_v, $\mathbf{b}_v$ and $\mathbf{p}_v$ to node v;
5) **for each** outcome j of the partition $\mathcal{P}$
   a) let D_j be the set of data in D satisfying outcome j;
   b) attach the node returned by **ML-tree**(D_j) to node v;
6) return v.

TABLE II
PSEUDOCODE OF THE SPLITTING TREE ALGORITHM

Splitting(D, $\mathbf{b}_v$)

Input: D, the current training instances;
$\mathbf{b}_v = (b_v(1), ..., b(q))$, the PLV on current node.
Output: a partition $\{D_1, D_2, ..., D_k\}$ for D.

1) **for** $l = 1$ to q
   a) **if** $b_v(l) = 0$, then learn a one-against-all SVM classifier for label l.
2) **for each** instance $x \in D$
   a) apply the set of trained SVM classifiers to x, if x is classified to the j-th class, then x is partitioned to j-th branch D_j accordingly.
3) return $\{D_1, D_2, ..., D_k\}$.

TABLE III
PSEUDOCODE OF THE STOPPING TREE ALGORITHM

Stopping(D, $\mathbf{b}_v$)

Input: D, the current training instances;
$\mathbf{b}_v = (b_v(1), ..., b(q))$, the PLV on current node.
Output: a boolean result, TRUE or FALSE

1) **if** all instances in D have the same labels, then return TRUE;
2) **if** all components of $\mathbf{b}_v = (b_v(1), ..., b_v(q))$ have values 1, then return TRUE;
3) **if** the classifier h_v has only one outcome, then return TRUE;
4) **otherwise**, return FALSE.

In this case, on one hand, the label of the parent node B (i.e., class y_2) is automatically transmitted to its child nodes B_1 and B_2. On the other hand, we can examine the purity of the classes, that is, $\mathbf{p}$, for the determination of PLVs. In this example, B_1 has large purity with respect to class y_1 (i.e., $p(1) = 1.0$ at the node B_1), while B_2 has large purity with respect to class y_3 [i.e., $p(3) = 1.0$ at the node B_2]. Therefore, B_1 is further labeled with class y_1, and B_2 is further labeled with y_3.

B. Learning Algorithm

The proposed ML-Tree algorithm is shown in Tables I–III. ML-Tree follows the divide-and-conquer paradigm of

algorithm design to construct a hierarchical tree. The algorithm takes a set of training instances D as input. The main loop (Table I) starts from a node v corresponding to the data D. Then we compute the values of $\mathbf{p}_v$ and $\mathbf{b}_v$ for node v, and find the best partition $\mathcal{P}$ for D by calling the splitting function in Table II. If such a partition can be found and the stopping criteria given in Table III are not satisfied, the algorithm will call itself recursively to create a new node for each outcome of the partition $\mathcal{P}$. If any of the stopping criteria is satisfied, the algorithm creates a leaf and computes the corresponding CPV and PLV.

The ML-Tree algorithm uses the splitting function in Table II partition the data D. First, a set of one-against-all SVM classifiers is learnt at the node, each one dealing with a class l of a nonpredictive label, that is, $b_v(l) = 0$ and $p_v(l) > 0$. Here, the PLV is used to select the individual one-against-all SVM classifiers for learning at each node. Specifically, we employ linear SVM as the base-classifier in this paper. After the construction of the SVM classifiers, each training instance in current node assigned a label corresponding to a child node (see Fig. 1) by the SVMs with the winner-takes-all strategy that an instance is classified to one class with maximum confidence score over the individual classifiers. If the scores of two (or more) of the classifiers are equally maximal, we classify the instance to the class that has maximal prior probability. The prior of the classes to node v is given in $\mathbf{p}_v$.

The ML-Tree algorithm uses three stopping criteria in Table III to determine when to stop growing the tree. They include examining the purity of the classes, the values of the predictive labels, and the outputs of the trained classifiers. Specifically, the first stopping criterion tests the purity of the instances in a node. If all instances at the node are labeled with the same single label or the same combination of labels, it is a leaf node.

Stopping Criterion 1: A node is a leaf node if all the instances at the node are pure.

Another way is to test the predictive label indicator vector $\mathbf{b}_v$ of the node v. If all the components of $\mathbf{b}_v$ are with value 1, the node is a leaf node.

Stopping Criterion 2: A node is a leaf node if all the classes are identified as predictive labels.

We can check the stopping criteria 1 and 2 before learning h_v for the node v. If the first two stopping criteria are satisfied, we stop growing the tree at that node. Otherwise, the proposed algorithm continue to train one-against-all SVM classifiers to partition the instances into different child nodes accordingly. If all the instances are partitioned into one child node, the node is a leaf node.

Stopping Criterion 3: A node is a leaf node if the data cannot be partitioned any further using the classifiers.

C. Multilabel Prediction

A algorithm outputs a prediction vector $\mathbf{h}_x = (h_x(1), h_x(2), \ldots, h_x(q))$ with boolean values is used for multilabel prediction to indicate the class membership of an unseen instance x, and a vector $\mathbf{s}_x = (s_x(1), s_x(2), \ldots, s_x(q))$ with real values to quantify the confidence that the instance x belongs to the predicted classes can be obtained from the tree. The prediction procedure starts with the one-against-all SVM classifiers h_{root} of the root node and follows a recursive process navigating an unseen instance x to h_v of a child node v based on the winner-takes-all decision rule. Eventually, the unseen instance x traverses to a leaf node v, and then x is assigned to the predictive labels of that node, that is, $h_x(l) = b_v(l)$. To make a relevant confidence estimation for x, we consider a scoring vector $\mathbf{r}_x = (r_x(1), r_x(2), \ldots, r_x(q))$ which is defined as the integration of the CPV along a decision-path from the root to the leaf node where x ends. Let $\mathcal{V}$ be a set of nodes in the decision-path, $v \in \mathcal{V}$ be a node in the path, and $\mathbf{p}_v$ be the CPV of node v where $p_v(l)$ is the lth component of $\mathbf{p}_v$. We define $r_x(l) = \sum_{v \in \mathcal{V}} p_v(l)$ and normalize $r_x(l)$ by $\mathrm{s_x}(l) = r_{\mathrm{x}}(l) / \sum_{l=1}^{q} r_{\mathrm{x}}(l)$. The resulting $s_x(l)$ indicates the confidence of label l for x.

D. Tuning the Parameter λ for Multilabel Prediction

ML-Tree algorithm constructs a hierarchical tree model using one-against-all SVM classifiers at each node for multilabel prediction. The prediction at each node of ML-Tree is based on the threshold parameter λ in (2). The value of λ is learnt based on cross-validation approach on the training set. In the experiments, we evaluate the value of λ between 0.5 and 1 with an increment of 0.05. When tuning the λ value best suited to a specific data set, a whole tree is grown using each λ value. The best λ value to minimize the hamming loss function of the constructed tree model is selected.

We note that it is desirable to recompute the best λ value after each split of the data, as it is likely that we have a new data distribution after a split. Recomputing λ is more appropriate than using fixed λ approach in the data sets with drastic changes in data distribution as the splitting continues. However, the computational time of recomputing λ is considerably high. Suppose we have K nodes for the whole tree, and λ has 10 possible values, then the number of λ evaluated in the cross-validation analysis is 10^K. The complexity is exponential with respect to the number of nodes in the tree. Such time cost is not practical for multilabel learning applications. Hence, we do not recompute λ in our tree model.

In our paper, we investigate the performance of our hierarchical tree method using fixed λ optimally tuned by cross validation on the training set. The experimental results show that our method is competitive against other multilabel learning methods on 11 multilabel data sets (see Section IV for detailed analysis).

E. Computational Complexity

Given a training data set containing m instances that are collected from the label space with q possible labels, the resulting multilabel learning problem is cast into a series of binary classification problems at each node. The complexity at each node depends on the number of learning labels that are required to construct classifiers to obtain the child nodes branch-out. Given that the number of the branches at the root node is q, in the worst scenario, the nodes in the second layer

would have $q-1$ branches, $q-2$ in the third layer, and so on. Hence, we have $q!$ terminal nodes. The total number of nodes in the tree will be $1+q+q(q-1)+q(q-1)(q-2)+\cdots+q!$, leading to the complexity of $O(f(m)(e-1)q!)$, where $f(m)$ is the complexity of the SVM classifier at each node with respect to the number of instances m. The current state of the art algorithms for training SVM classifier have a time complexity scaling close to $O(m^2)$ [22].

To simplify the analysis, we assume that the hierarchy is a complete k-ary tree of depth d reported in [23], that is, the average branch-out is k and the average depth of the tree is d. It indicates that all internal nodes have k child nodes and all terminal nodes are at the bottom layer. The cost of one SVM classifier is $f(|D_v|) \approx O(|D_v|^2)$ with respect to node v, and we have k classifiers at the node, that is, $O(k|D_v|^2)$, then the cost at the root node is km^2, while at the second layer we have k additional cost of $k(m/k)^2$, and k^2 additional cost of $k(m/k^2)^2$ at the third layer, and so on. Thus, $k(m^2 + m^2/k + m^2/k^2 + \cdots + m^2/k^p)$ for learning all the classifiers in the tree. This leads to a total cost of $O(2km^2)$ as a sum of a geometric series when the depth of the tree p approaches infinity.

In general, the overall complexity of ML-Tree depends on the number of instances and the average number of branch-outs. In real world applications, most of the nodes become terminal at very early stage or have much lower number of branch-outs. The nodes at the lower levels of the tree train their classifiers using only $|D_v|$ samples at the node v ($|D_v| < m$). On the other hand, the labels identified as predictive labels do not need further processing when moving down the tree. We expect the number of branch-outs in the child nodes to be smaller than those of their parent nodes. So the complexity of training the classifiers at the lower levels of the tree is expected to fall. In fact, an accurate complexity analysis is hard to achieve, because it depends on many factors. One of the key factors is the average number of labels per example (label cardinality [13]). The larger this statistic is, the higher the expected complexity of growing a tree will be. For empirical analysis of the complexity of growing a tree on data sets with respect to label cardinality, we refer to Section IV-G.

IV. Experiments

A. Data Sets

We use 11 multilabel data sets in our experimental study. These data sets are benchmark data sets from different application domains. The data are originally split into training and testing set. We use the training and test split in the experiments. The characteristics of the data sets are summarized in Table IV. The multilabel data sets are available at http://mulan.sourceforge.net/datasets.html.

The data sets are from various domains, including biology, multimedia and text categorization. Reuters21578 consists of three different data sets: Reuters(10), Reuters(21), and Reuters(90). These three data sets are the subsets of 10, 21, and 90 classes from the Reuters21578 collection. The Ohsumed is a data subset of the 24 MeSH disease categories. For more information about the data sets, please refer to [33].

TABLE IV
DESCRIPTION OF THE MULTILABEL DATA SETS IN TERMS OF NUMBER OF TRAINING (#TR.E.) AND TEST (#T.E.) EXAMPLES, THE NUMBER OF FEATURES (d), THE TOTAL NUMBER OF LABELS (q), AND CARDINALITY (l_c). THE PROBLEMS ARE ORDERED BY THEIR COMPLEXITY CALCULATED AS #TR.E. $\times d \times q$

Dataset	domain	#tr.e.	#t.e.	d	q	l_c
emotions	Multimedia	391	202	72	6	1.87
scene	Multimedia	1,211	1,196	294	6	1.07
yeast	Biology	1,500	917	103	14	4.24
medical	Text	333	645	1,449	45	1.25
Reuters(10)	Text	6,490	2,545	500	10	1.11
Reuters(21)	Text	7,140	2,747	500	21	1.16
ohsumed	Text	6,286	7,643	500	24	1.66
tmc2007	Text	21,519	7,077	500	22	2.16
Reuters(90)	Text	7,770	3,019	500	90	1.24
corel5k	Multimedia	4,500	500	499	374	3.52
bibtex	Text	4,880	2,515	1,836	159	2.40

B. Compared Algorithms and Parameter Instantiation

In this paper, we compare the ML-Tree method with six multilabel algorithms, namely binary relevance (BR), classifier chain (CC) [34], multilabel k nearest neighbor (ML-kNN), IBLR-ML, HOMER, and PCT. Our comparative study is based on the MULAN library [35][1] and the CLUS system.[2] The MULAN library is used for BR, CC, ML-kNN, IBLR-ML, and HOMER. The CLUS system is used for PCT. The experiments are conducted on a AMD 3.4 GHz machine with 64 GB RAM running Windows Server 2008. BR is a problem transformation method, CC involves binary classifiers linked along a chain. HOMER constructs a hierarchy of the multiple labels and a classifier for the label sets in each node of the hierarchy. Our proposed ML-Tree method uses a base-classifier at each node in the tree generation. These methods use base-classifiers for solving the partial binary classification problems. For fair comparison, SVM with linear kernel is used as the base-classifier for BR, CC, HOMER, and ML-Tree. We adopt the implementation in the LIBSVM library [36] to learn SVM classifiers.

In the experiments, the parameters of the learning algorithms are tuned in the experiments using fivefold cross validation on the training set. The parameter values of the algorithms yielding the best average hamming loss in the fivefold cross validation are chosen. After determining the best parameters for each method on every data set, the classifiers are trained with the best parameters on the whole training set and evaluated on the corresponding testing set. We describe the parameter settings of the learning algorithms as follows. The number of neighbors k in the ML-kNN and IBLR-ML is determined from 5 to 30 with an increment of 5. The number of clusters in the HOMER is determined in the range of 2-6. The values considered for penalty C in SVMs of BR, CC, HOMER, and ML-Tree are $2^{-5}, 2^{-3}, \ldots, 2^3$. The exception to this is the tmc2007 data set where the penalty C is simply set to 2^{-5} because of its high computational complexity. The prediction at each node of ML-Tree is based on a

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

TABLE V
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF HAMMING LOSS ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.2137(2)	0.2368(4)	0.2830(7)	0.2797(6)	**0.2112(1)**	0.2591(5)	0.2343(3)
scene	0.1021(4)	0.1054(5)	0.0960(2)	**0.0893(1)**	0.1180(6)	0.1332(7)	0.0991(3)
yeast	0.0165(2)	0.0166(3)	0.0218(6)	0.0208(5)	**0.0155(1)**	0.2163(7)	0.0175(4)
medical	**0.0118(1)**	0.0120(2)	0.0197(5)	0.0299(7)	0.0141(4)	0.0230(6)	0.0134(3)
Reuters(10)	0.0161(2)	0.0165(3)	0.0218(6)	0.0208(5)	**0.0155(1)**	0.0288(7)	0.0175(4)
Reuters(21)	**0.0107(1)**	0.0109(2)	0.0141(6)	0.0130(5)	0.0115(3)	0.0214(7)	0.0127(4)
ohsumed	**0.0449(1)**	0.0453(2)	0.0564(5)	0.0569(6)	0.0480(3)	0.0624(7)	0.0522(4)
tmc2007	0.0548(2)	0.0552(3)	0.0611(6)	0.0583(5)	0.0578(4)	0.0759(7)	**0.0506(1)**
Reuters(90)	0.0051(2)	**0.0049(1)**	0.0053(4)	0.0058(6)	0.0051(3)	0.0118(7)	0.0057(5)
corel5k	**0.0092(1)**	0.0093(2)	0.0093(3)	0.0231(7)	0.0103(6)	0.0094(4)	0.0100(5)
bibtex	**0.0123(1)**	0.0125(2)	0.0139(4)	0.0189(7)	0.0137(3)	0.0145(5)	0.0170(6)
Avg. rank	**1.727**	2.636	4.909	5.455	3.182	6.273	3.818

TABLE VI
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF ACCURACY ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.517(2)	**0.561(1)**	0.318(6)	0.316(7)	0.492(3)	0.483(4)	0.468(5)
scene	0.576(6)	0.682(2)	0.624(4)	0.649(3)	0.584(5)	0.570(7)	**0.714(1)**
yeast	0.894(4)	0.904(3)	0.860(6)	0.871(5)	0.905(2)	0.443(7)	**0.918(1)**
medical	0.707(3)	0.719(2)	0.373(6)	0.422(5)	0.671(4)	0.204(7)	**0.720(1)**
Reuters(10)	0.904(4)	0.905(3)	0.860(7)	0.871(5)	0.905(2)	0.860(6)	**0.918(1)**
Reuters(21)	0.878(3)	0.879(2)	0.855(5)	0.851(6)	0.873(4)	0.779(7)	**0.887(1)**
ohsumed	0.469(3)	0.458(4)	0.324(5)	0.276(6)	**0.529(1)**	0.188(7)	0.528(2)
tmc2007	0.614(2)	**0.615(1)**	0.557(6)	0.565(5)	0.602(4)	0.454(7)	0.613(3)
Reuters(90)	0.783(5)	0.787(3)	0.779(6)	0.786(4)	0.797(2)	0.568(7)	**0.797(1)**
corel5k	0.052(4)	0.055(3)	0.018(6)	0.041(5)	**0.106(1)**	0.000(7)	0.058(2)
bibtex	0.319(3)	**0.340(1)**	0.129(6)	0.165(5)	0.335(2)	0.046(7)	0.287(4)
Avg. rank	3.545	2.273	5.727	5.091	2.727	6.636	**2**

TABLE VII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF PRECISION ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.623(3)	0.619(4)	0.511(6)	0.494(7)	0.634(2)	0.582(5)	**0.692(1)**
scene	0.601(6)	0.712(2)	0.655(4)	0.680(3)	0.613(5)	0.599(7)	**0.749(1)**
yeast	0.907(4)	0.916(3)	0.873(6)	0.882(5)	0.917(2)	0.711(7)	**0.939(1)**
medical	0.748(3)	0.757(2)	0.416(6)	0.467(5)	0.718(4)	0.257(7)	**0.822(1)**
Reuters(10)	0.916(4)	0.917(2)	0.873(6)	0.882(5)	0.917(3)	0.873(7)	**0.939(1)**
Reuters(21)	0.897(3)	0.899(2)	0.877(5)	0.871(6)	0.890(4)	0.813(7)	**0.924(1)**
ohsumed	0.636(4)	**0.663(1)**	0.434(5)	0.371(6)	0.647(3)	0.266(7)	0.655(2)
tmc2007	0.762(3)	0.764(2)	0.724(6)	0.728(5)	0.734(4)	0.648(7)	**0.815(1)**
Reuters(90)	0.812(5)	0.818(3)	0.811(6)	0.816(4)	0.825(2)	0.579(7)	**0.836(1)**
corel5k	0.133(2)	0.116(3)	0.043(6)	0.068(5)	**0.215(1)**	0.000(7)	0.080(4)
bibtex	0.489(2)	**0.490(1)**	0.254(5)	0.253(6)	0.477(3)	0.140(7)	0.394(4)
Avg. rank	3.545	2.273	5.545	5.182	3	6.818	**1.636**

TABLE VIII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF RECALL ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.609(2)	**0.740(1)**	0.364(6)	0.362(7)	0.537(4)	0.584(3)	0.488(5)
scene	0.610(6)	0.709(2)	0.649(4)	0.658(3)	0.635(5)	0.571(7)	**0.740(1)**
yeast	0.911(4)	0.918(3)	0.875(6)	0.885(5)	0.924(2)	0.490(7)	**0.928(1)**
medical	0.761(2)	**0.772(1)**	0.395(6)	0.534(5)	0.719(4)	0.204(7)	0.750(3)
Reuters(10)	0.925(2)	0.919(4)	0.875(6)	0.885(5)	0.924(3)	0.871(7)	**0.928(1)**
Reuters(21)	**0.905(1)**	0.898(3)	0.874(5)	0.867(6)	0.897(4)	0.782(7)	0.902(2)
ohsumed	0.546(3)	0.572(2)	0.358(5)	0.296(6)	**0.623(1)**	0.188(7)	0.545(4)
tmc2007	0.687(3)	0.688(2)	0.654(6)	0.655(5)	**0.738(1)**	0.532(7)	0.683(4)
Reuters(90)	0.809(3)	0.805(4)	0.796(6)	0.803(5)	**0.823(1)**	0.568(7)	0.815(2)
corel5k	0.054(5)	0.059(4)	0.019(6)	0.077(2)	**0.123(1)**	0.000(7)	0.071(3)
bibtex	0.337(4)	0.366(3)	0.134(6)	0.209(5)	**0.380(1)**	0.046(7)	0.369(2)
Avg. rank	3.182	2.636	5.636	4.909	**2.455**	6.636	2.545

TABLE IX
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF F1 SCORE ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.590(2)	**0.649(1)**	0.396(6)	0.391(7)	0.558(3)	0.556(4)	0.549(5)
scene	0.596(6)	0.701(2)	0.643(4)	0.662(3)	0.610(5)	0.580(7)	**0.735(1)**
yeast	0.904(4)	0.913(3)	0.869(6)	0.879(5)	0.916(2)	0.550(7)	**0.929(1)**
medical	0.739(3)	0.750(2)	0.395(6)	0.473(5)	0.704(4)	0.221(7)	**0.764(1)**
Reuters(10)	0.915(3)	0.914(4)	0.869(6)	0.879(5)	0.916(2)	0.868(7)	**0.929(1)**
Reuters(21)	0.893(2)	0.892(3)	0.869(5)	0.863(6)	0.887(4)	0.791(7)	**0.904(1)**
ohsumed	0.560(3)	0.585(2)	0.370(5)	0.313(6)	**0.602(1)**	0.211(7)	0.557(4)
tmc2007	0.683(4)	0.684(3)	0.647(6)	0.651(5)	0.696(2)	0.544(7)	**0.699(1)**
Reuters(90)	0.801(4)	0.803(3)	0.795(6)	0.801(5)	0.815(2)	0.571(7)	**0.816(1)**
corel5k	0.073(3)	0.073(2)	0.025(6)	0.061(5)	**0.144(1)**	0.000(7)	0.073(4)
bibtex	0.373(3)	0.394(2)	0.162(6)	0.209(5)	**0.394(1)**	0.067(7)	0.351(4)
Avg. rank	3.364	2.455	5.636	5.182	2.455	6.727	**2.182**

threshold parameter λ. We tune λ from 0.5 to 1 with an increment of 0.05. When tuning the parameter λ and C for ML-Tree, the whole tree is grown by using the evaluated parameter values at each node. We then select the best parameter values to optimize the hamming loss performance of the constructed tree model. Multilabel predictions of BR and CC can also be made from a threshold parameter t. Specifically, given a vector $\mathbf{w}_i \in \mathbb{R}^q$ of real-valued confidence outputs for an unlabeled instance x_i, a multilabel prediction $\hat{Y}_i = \{\hat{Y}_{i,1}, \hat{Y}_{i,2}, \ldots, \hat{Y}_{i,q}\}$ of x_i can be obtained under a threshold function such that: $\hat{Y}_{i,j} = 1$ if $w_{i,j} \geq t$, and $\hat{Y}_{i,j} = 0$ otherwise [34]. In the experiments, the threshold t is tuned in the range from 0.5 to 1 with an interval of 0.05 for BR and CC. We find that this threshold selection method is more effective than using an arbitrary threshold, for example, 0.5. Note that this setup relies on the output of probability estimates of SVM classifiers. The one-against-all SVM with probability output has been implemented in the LIBSVM library. We set the LIBSVM with prediction options −b1 to learn SVM classifiers with probability outputs.

C. Performance Measures and Statistical Evaluation

With respect to the outputs of learning algorithms, the evaluation measures can be grouped into two categories: bipartition-based and ranking-based. The bipartition-based measures are based on the comparison of the predicted predictive labels with the ground-truth predictive labels. The bipartition-based measures can be further divided into example-based and label-based. The example-based measures are based on the average difference between the actual and the predicted set of labels over the examples. The label-based measures are based on the predictive performance for each label separately such that the performance over all labels are averaged. The ranking-based measures compare the predicted ranking of the label with the ground truth ranking. In our experiments, we use six example-based evaluation measures (Hamming loss, accuracy, precision, recall, subset accuracy, and F1 score), six label-based evaluation measures (microprecision, microrecall, microF1 score, macroprecision, macrorecall, and macroF1 score) and four ranking-based measures (one-error, coverage, ranking loss, and average precision). A detailed description of these evaluation measures can be found in [33].

As multiple algorithms are compared on multiple data sets, we follow the two-step statistical test procedure recommended by Demšar [37] to analyze the experimental results. The test procedure consists of a Friedman test of the null hypothesis that all learners have equal performance and a Nemenyi test to compare learners in a pairwise way. All tests are based on the average ranks shown in the bottom row of Tables V–XXII. The results with respect to the post hoc tests are visually presented with diagrams (as suggested in [37]) in Figs. 2–4. The horizontal axis with value $1 - x$ shows the average rank of a method. The lowest(best) rank is at the right-most side of the axis. The algorithms that do not differ significantly (at the significantly from each other level of

TABLE X
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF SUBSET ACCURACY ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	**0.292(1)**	0.282(2)	0.104(6)	0.104(7)	0.277(3)	0.262(4)	0.223(5)
scene	0.518(6)	0.625(2)	0.569(4)	0.611(3)	0.505(7)	0.539(5)	**0.652(1)**
yeast	0.863(4)	0.876(2)	0.830(6)	0.843(5)	0.874(3)	0.142(7)	**0.884(1)**
medical	0.614(2)	**0.628(1)**	0.307(5)	0.285(6)	0.574(4)	0.154(7)	0.591(3)
Reuters(10)	0.870(4)	0.876(2)	0.830(7)	0.843(5)	0.874(3)	0.835(6)	**0.884(1)**
Reuters(21)	0.832(3)	**0.840(1)**	0.813(5)	0.813(6)	0.830(4)	0.743(7)	0.837(2)
ohsumed	0.310(3)	0.306(4)	0.197(5)	0.175(6)	0.319(2)	0.127(7)	**0.319(1)**
tmc2007	0.314(3)	0.324(2)	0.282(6)	0.296(5)	0.305(4)	0.203(7)	**0.349(1)**
Reuters(90)	0.727(6)	0.738(4)	0.731(5)	0.741(3)	0.744(2)	0.559(7)	**0.744(1)**
corel5k	0.006(3)	0.006(4)	0.000(6)	0.002(5)	**0.018(1)**	0.000(7)	0.014(2)
bibtex	0.183(2)	**0.199(1)**	0.055(6)	0.071(5)	0.183(3)	0.004(7)	0.136(4)
Avg. rank	3.364	2.273	5.545	5.091	3.273	6.455	**2**

TABLE XI
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MICRO PRECISION ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.698(3)	0.616(6)	0.625(5)	0.633(4)	**0.751(1)**	0.610(7)	0.717(2)
scene	0.787(3)	0.715(5)	0.796(2)	**0.830(1)**	0.695(6)	0.657(7)	0.729(4)
yeast	**0.947(1)**	0.939(3)	0.936(5)	0.939(4)	0.942(2)	0.705(7)	0.930(6)
medical	0.812(2)	0.795(4)	0.775(5)	0.462(7)	0.765(6)	**0.834(1)**	0.796(3)
Reuters(10)	0.937(4)	0.940(2)	0.936(5)	0.939(3)	**0.942(1)**	0.872(7)	0.930(6)
Reuters(21)	0.918(2)	**0.924(1)**	0.890(6)	0.917(3)	0.910(4)	0.842(7)	0.905(5)
ohsumed	0.812(2)	**0.827(1)**	0.705(5)	0.753(3)	0.680(6)	0.733(4)	0.628(7)
tmc2007	0.738(3)	0.737(4)	0.735(5)	0.757(2)	0.717(6)	0.660(7)	**0.805(1)**
Reuters(90)	0.850(3)	**0.868(1)**	0.854(2)	0.809(6)	0.835(4)	0.588(7)	0.816(5)
corel5k	0.616(2)	0.575(3)	**0.735(1)**	0.049(6)	0.366(4)	0.000(7)	0.360(5)
bibtex	0.764(3)	0.706(4)	0.832(2)	0.316(7)	0.594(5)	**1.000(1)**	0.430(6)
Avg. rank	**2.545**	3.091	3.909	4.182	4.091	5.636	4.545

TABLE XII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MICRO RECALL ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.619(2)	**0.744(1)**	0.351(7)	0.358(6)	0.536(4)	0.589(3)	0.476(5)
scene	0.597(6)	0.694(2)	0.631(4)	0.637(3)	0.621(5)	0.553(7)	**0.720(1)**
yeast	0.899(4)	0.907(3)	0.860(6)	0.867(5)	**0.915(1)**	0.495(7)	0.909(2)
medical	0.745(2)	**0.760(1)**	0.401(6)	0.515(5)	0.705(3)	0.207(7)	0.691(4)
Reuters(10)	0.914(2)	0.907(4)	0.860(7)	0.867(5)	**0.915(1)**	0.864(6)	0.909(3)
Reuters(21)	**0.883(1)**	0.874(3)	0.847(5)	0.840(6)	0.877(2)	0.751(7)	0.859(4)
ohsumed	0.460(3)	0.440(4)	0.322(5)	0.268(6)	**0.581(1)**	0.159(7)	0.527(2)
tmc2007	**0.701(1)**	0.696(3)	0.610(6)	0.614(5)	0.698(2)	0.498(7)	0.652(4)
Reuters(90)	0.767(2)	0.760(3)	0.746(6)	0.756(4)	**0.783(1)**	0.487(7)	0.756(5)
corel5k	0.056(5)	0.061(4)	0.020(6)	0.079(2)	**0.127(1)**	0.000(7)	0.074(3)
bibtex	0.288(4)	0.320(3)	0.121(6)	0.195(5)	**0.337(1)**	0.057(7)	0.322(2)
Avg. rank	2.909	2.818	5.818	4.727	**2**	6.545	3.182

TABLE XIII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MICROF1 ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.656(2)	**0.674(1)**	0.449(7)	0.458(6)	0.626(3)	0.600(4)	0.572(5)
scene	0.679(5)	0.705(3)	0.704(4)	0.721(2)	0.656(6)	0.600(7)	**0.725(1)**
yeast	0.923(3)	0.923(2)	0.896(6)	0.901(5)	**0.928(1)**	0.582(7)	0.919(4)
medical	**0.777(1)**	0.777(2)	0.529(5)	0.487(6)	0.734(4)	0.332(7)	0.740(3)
Reuters(10)	0.925(2)	0.923(3)	0.896(6)	0.901(5)	**0.928(1)**	0.868(7)	0.919(4)
Reuters(21)	**0.900(1)**	0.898(2)	0.868(6)	0.876(5)	0.893(3)	0.794(7)	0.881(4)
ohsumed	0.587(2)	0.574(3)	0.442(5)	0.396(6)	**0.627(1)**	0.262(7)	0.573(4)
tmc2007	0.719(2)	0.716(3)	0.666(6)	0.678(5)	0.707(4)	0.568(7)	**0.721(1)**
Reuters(90)	0.806(3)	**0.811(1)**	0.796(4)	0.782(6)	0.808(2)	0.532(7)	0.785(5)
corel5k	0.102(4)	0.111(3)	0.040(6)	0.060(5)	**0.188(1)**	0.000(7)	0.122(2)
bibtex	0.418(3)	**0.440(1)**	0.212(6)	0.241(5)	0.430(2)	0.108(7)	0.368(4)
Avg. rank	2.545	**2.182**	5.545	5.091	2.545	6.727	3.364

TABLE XIV
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MACRO PRECISION ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.684(2)	0.606(5)	0.475(7)	0.477(6)	**0.705(1)**	0.619(4)	0.677(3)
scene	0.789(3)	0.727(5)	0.793(2)	**0.831(1)**	0.694(6)	0.656(7)	0.740(4)
yeast	**0.912(1)**	0.908(2)	0.874(5)	0.881(4)	0.901(3)	0.488(7)	0.865(6)
medical	0.342(2)	0.324(3)	0.148(6)	0.202(5)	0.313(4)	0.018(7)	**0.404(1)**
Reuters(10)	0.902(2)	**0.907(1)**	0.874(5)	0.881(4)	0.901(3)	0.825(7)	0.865(6)
Reuters(21)	0.855(3)	**0.864(1)**	0.823(5)	0.858(2)	0.829(4)	0.649(7)	0.798(6)
ohsumed	0.732(2)	**0.737(1)**	0.633(4)	0.652(3)	0.629(5)	0.119(7)	0.579(6)
tmc2007	0.757(3)	0.738(4)	0.738(5)	0.783(2)	0.695(6)	0.395(7)	**0.856(1)**
Reuters(90)	0.555(3)	0.564(2)	0.543(4)	0.450(6)	**0.577(1)**	0.034(7)	0.517(5)
corel5k	0.051(4)	0.052(3)	0.033(5)	0.033(6)	0.055(2)	0.000(7)	**0.063(1)**
bibtex	**0.501(1)**	0.491(2)	0.194(5)	0.170(6)	0.431(3)	0.006(7)	0.337(4)
Avg. rank	**2.364**	2.636	4.818	4.091	3.455	6.727	3.909

TABLE XV
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MACRO RECALL ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.605(2)	**0.736(1)**	0.313(7)	0.328(6)	0.529(4)	0.585(3)	0.465(5)
scene	0.602(6)	0.697(2)	0.637(4)	0.644(3)	0.627(5)	0.561(7)	**0.727(1)**
yeast	0.824(3)	0.837(2)	0.754(5)	0.742(6)	**0.857(1)**	0.269(7)	0.823(4)
medical	0.313(2)	**0.323(1)**	0.086(6)	0.214(5)	0.256(4)	0.022(7)	0.312(3)
Reuters(10)	0.847(2)	0.838(3)	0.754(6)	0.742(7)	**0.857(1)**	0.806(5)	0.823(4)
Reuters(21)	**0.773(1)**	0.754(3)	0.698(4)	0.695(5)	0.756(2)	0.544(7)	0.693(6)
ohsumed	0.374(3)	0.374(4)	0.236(5)	0.209(6)	**0.531(1)**	0.082(7)	0.460(2)
tmc2007	0.536(4)	0.548(3)	0.407(6)	0.427(5)	0.561(2)	0.260(7)	**0.655(1)**
Reuters(90)	0.413(2)	0.395(4)	0.355(6)	0.406(3)	**0.443(1)**	0.037(7)	0.385(5)
corel5k	0.014(5)	0.022(4)	0.009(6)	**0.043(1)**	0.031(2)	0.000(7)	0.026(3)
bibtex	0.208(4)	0.239(3)	0.051(6)	0.118(5)	**0.260(1)**	0.006(7)	0.253(2)
Avg. rank	3.091	2.727	5.545	4.727	**2.182**	6.455	3.273

TABLE XVI
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF MACROF1 ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.607(2)	**0.662(1)**	0.347(7)	0.371(6)	0.578(4)	0.596(3)	0.522(5)
scene	0.682(5)	0.711(3)	0.700(4)	0.723(2)	0.657(6)	0.603(7)	**0.729(1)**
yeast	0.864(3)	0.869(2)	0.805(5)	0.800(6)	**0.877(1)**	0.290(7)	0.840(4)
medical	0.312(3)	0.317(2)	0.104(6)	0.181(5)	0.269(4)	0.020(7)	**0.331(1)**
Reuters(10)	0.872(2)	0.870(3)	0.805(6)	0.800(7)	**0.877(1)**	0.813(5)	0.840(4)
Reuters(21)	**0.809(1)**	0.801(2)	0.741(5)	0.761(4)	0.788(3)	0.558(7)	0.731(6)
ohsumed	0.473(4)	0.478(3)	0.317(5)	0.305(6)	**0.544(1)**	0.096(7)	0.502(2)
tmc2007	0.598(4)	0.607(2)	0.485(6)	0.521(5)	0.605(3)	0.294(7)	**0.727(1)**
Reuters(90)	0.459(2)	0.449(3)	0.408(6)	0.409(5)	**0.475(1)**	0.034(7)	0.418(4)
corel5k	0.020(5)	0.026(4)	0.013(6)	0.029(3)	0.033(2)	0.000(7)	**0.034(1)**
bibtex	0.262(4)	0.291(2)	0.069(6)	0.131(5)	**0.301(1)**	0.006(7)	0.273(3)
Avg. rank	3.182	**2.455**	5.636	4.909	**2.455**	6.455	2.909

TABLE XVII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF ONE ERROR ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	**0.292(1)**	0.322(3)	0.361(6)	0.356(5)	0.351(4)	0.381(7)	0.302(2)
scene	0.242(2)	0.268(5)	0.253(4)	**0.234(1)**	0.329(6)	0.381(7)	0.248(3)
yeast	0.053(2)	0.053(3)	0.090(6)	0.086(5)	0.073(4)	0.251(7)	**0.053(1)**
medical	**0.171(1)**	0.174(2)	0.361(5)	0.526(6)	0.247(4)	0.628(7)	0.206(3)
Reuters(10)	**0.053(1)**	0.054(2)	0.090(6)	0.086(5)	0.073(4)	0.122(7)	0.055(3)
Reuters(21)	**0.064(1)**	0.065(2)	0.101(6)	0.091(4)	0.096(5)	0.174(7)	0.066(3)
ohsumed	0.227(3)	0.222(2)	0.389(5)	0.412(6)	0.320(4)	0.539(7)	**0.216(1)**
tmc2007	0.171(2)	0.174(3)	0.204(5)	0.195(4)	0.237(6)	0.298(7)	**0.170(1)**
Reuters(90)	0.143(2)	**0.141(1)**	0.153(4)	0.161(5)	0.165(6)	0.419(7)	0.146(3)
corel5k	0.674(2)	0.686(3)	0.694(4)	0.882(7)	0.758(5)	0.768(6)	**0.672(1)**
bibtex	**0.357(1)**	0.359(2)	0.602(5)	0.629(6)	0.458(4)	0.783(7)	0.360(3)
Avg. rank	**1.636**	2.545	5.091	4.909	4.727	6.909	2.182

$p = 0.05$) are connected by a bold horizontal line.

D. Results on the Example-Based Measures

In this section, we present the results from the example-based evaluation measures, that is, hamming loss, accuracy, precision, recall, F1 score, and subset accuracy. The example-based measures are commonly used for multilabel classification. The comparative results of all the algorithms are shown in Tables V–X. The uparrow ↑ (downarrow ↓) indicates that a larger (smaller) value is more appreciated for a specific evaluation measure. The number in brackets is the rank of the method. For each data set (on each row of the tables), the methods are ranked in a descending order based on their performance. The numbers in boldface indicate the best ranking results. The last row in the table presents the average ranks across the problems. The graphical presentation results of the Nemenyi post hoc tests based on the average ranks are shown in Fig. 2.

In the experiment, we compare our proposed ML-Tree method with different multilabel learning methods (BR, CC, ML-kNN, IBLR-ML, HOMER, and PCT) and use a wide range of example-based measures to evaluate their

TABLE XVIII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF COVERAGE ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	1.946(2)	**1.901(1)**	2.515(6)	2.297(4)	2.663(7)	2.352(5)	2.055(3)
scene	**0.501(1)**	0.578(4)	0.533(2)	0.548(3)	1.120(7)	0.961(6)	0.628(5)
yeast	**0.202(1)**	0.206(2)	0.343(5)	0.332(4)	0.584(6)	6.666(7)	0.280(3)
medical	**2.119(1)**	2.347(2)	3.277(4)	5.549(5)	5.992(7)	5.898(6)	3.265(3)
Reuters(10)	**0.204(1)**	0.206(2)	0.343(5)	0.332(4)	0.584(7)	0.394(6)	0.294(3)
Reuters(21)	**0.361(1)**	0.363(2)	0.720(5)	0.564(3)	1.670(7)	0.943(6)	0.604(4)
ohsumed	2.627(2)	**2.487(1)**	4.192(3)	4.605(5)	9.892(7)	6.693(6)	4.402(4)
tmc2007	2.491(4)	2.571(5)	2.405(3)	**2.321(1)**	7.821(7)	4.135(6)	2.376(2)
Reuters(90)	2.096(2)	**2.082(1)**	2.925(3)	3.198(4)	12.741(7)	7.133(6)	3.198(5)
corel5k	**105.6(1)**	120.1(3)	111.2(2)	199.0(6)	320.5(7)	120.5(4)	123.4(5)
bibtex	**23.662(1)**	24.13(2)	61.05(6)	48.78(4)	74.12(7)	58.60(5)	37.48(3)
Avg. rank	**1.545**	2.273	4	3.909	6.909	5.727	3.636

TABLE XIX
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF RANKING LOSS ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.179(2)	**0.178(1)**	0.280(6)	0.257(4)	0.311(7)	0.265(5)	0.199(3)
scene	**0.080(1)**	0.096(4)	0.087(2)	0.089(3)	0.200(7)	0.172(6)	0.105(5)
yeast	**0.011(1)**	0.011(2)	0.026(5)	0.024(4)	0.050(6)	0.199(7)	0.018(3)
medical	**0.032(1)**	0.035(2)	0.054(4)	0.103(5)	0.103(6)	0.114(7)	0.053(3)
Reuters(10)	**0.011(1)**	0.011(2)	0.026(5)	0.024(4)	0.050(7)	0.032(6)	0.020(3)
Reuters(21)	**0.008(1)**	0.008(2)	0.022(5)	0.018(4)	0.064(7)	0.035(6)	0.018(3)
ohsumed	0.054(2)	**0.051(1)**	0.109(4)	0.124(5)	0.288(7)	0.204(6)	0.106(3)
tmc2007	0.036(3)	0.038(5)	0.038(4)	0.035(2)	0.183(7)	0.089(6)	**0.034(1)**
Reuters(90)	0.014(2)	**0.014(1)**	0.020(3)	0.024(5)	0.114(7)	0.065(6)	0.024(4)
corel5k	**0.118(1)**	0.133(3)	0.127(2)	0.253(6)	0.580(7)	0.141(4)	0.148(5)
bibtex	**0.078(1)**	0.079(2)	0.240(5)	0.196(4)	0.297(7)	0.256(6)	0.146(3)
Avg. rank	**1.455**	2.273	4.091	4.182	6.818	5.909	3.273

TABLE XX
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF AVERAGE PRECISION ↑

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	**0.786(1)**	0.782(2)	0.709(7)	0.731(4)	0.709(6)	0.721(5)	0.778(3)
scene	**0.856(1)**	0.838(5)	0.849(3)	0.856(2)	0.763(6)	0.755(7)	0.844(4)
yeast	**0.968(1)**	0.968(2)	0.941(5)	0.944(4)	0.937(6)	0.727(7)	0.963(3)
medical	**0.868(1)**	0.861(2)	0.729(5)	0.605(6)	0.758(4)	0.491(7)	0.809(3)
Reuters(10)	**0.968(1)**	0.967(2)	0.941(5)	0.944(4)	0.937(6)	0.923(7)	0.963(3)
Reuters(21)	**0.957(1)**	0.957(2)	0.927(5)	0.936(4)	0.908(6)	0.872(7)	0.947(3)
ohsumed	0.805(2)	**0.812(1)**	0.667(4)	0.642(5)	0.626(6)	0.500(7)	0.747(3)
tmc2007	0.850(2)	0.845(3)	0.828(5)	0.838(4)	0.724(6)	0.722(7)	**0.858(1)**
Reuters(90)	0.891(2)	**0.891(1)**	0.877(4)	0.875(5)	0.828(6)	0.658(7)	0.879(3)
corel5k	**0.291(1)**	0.276(3)	0.277(2)	0.162(6)	0.148(7)	0.216(5)	0.270(4)
bibtex	0.570(2)	**0.570(1)**	0.329(6)	0.335(5)	0.415(4)	0.212(7)	0.524(3)
Avg. rank	**1.364**	2.182	4.636	4.455	5.727	6.636	3

TABLE XXI
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF TRAINING TIME (IN SECONDS) ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	18 465(6)	13 576(5)	4(2)	6(3)	12 524(4)	**1(1)**	20 175(7)
scene	11 526(6)	9 793(4)	113(3)	64(2)	11 189(5)	**2(1)**	33 500(7)
yeast	25 783(6)	10 306(4)	552(2)	688(3)	15 115(5)	**2(1)**	63 297(7)
medical	382(4)	382(5)	4(2)	20(3)	585(6)	**2(1)**	1 887(7)
Reuters(10)	14 208(6)	11 707(4)	552(2)	681(3)	12 668(5)	**7(1)**	44 242(7)
Reuters(21)	27 712(6)	24 757(4)	673(3)	627(2)	27 018(5)	**18(1)**	64 235(7)
ohsumed	257 792(5)	271 809(6)	3 311(3)	2 603(2)	132 036(4)	**756(1)**	719 812(7)
tmc2007	143 719(6)	134 410(5)	9 649(3)	6 615(2)	48 661(4)	**50(1)**	249 247(7)
Reuters(90)	43 472(5)	40 366(4)	299(2)	2 202(3)	60 847(7)	**63(1)**	53 362(6)
corel5k	237 112(5)	265 995(6)	383(2)	40 033(3)	141 933(4)	**258(1)**	308 374(7)
bibtex	303 141(6)	302 025(5)	1 665(2)	9 141(3)	182 242(4)	**195(1)**	642 109(7)
Avg. rank	5.545	4.727	2.364	2.636	4.818	**1**	6.909

TABLE XXII
PERFORMANCE OF THE MULTILABEL LEARNING ALGORITHMS IN TERMS OF TESTING TIME (IN SECONDS) ↓

Dataset	BR	CC	MLkNN	IBLR-ML	HOMER	PCT	ML-Tree
emotions	0.78(7)	0.61(4)	0.30(2)	0.34(3)	0.77(6)	**0.27(1)**	0.75(5)
scene	14.07(5)	14.10(6)	5.29(3)	3.15(2)	17.59(7)	**0.08(1)**	13.22(4)
yeast	12.27(4)	8.57(2)	9.81(3)	13.59(5)	14.58(6)	**0.92(1)**	16.70(7)
medical	3.18(6)	3.14(5)	0.78(2)	**0.56(1)**	5.54(7)	1.48(3)	2.96(4)
Reuters(10)	5.09(3)	4.69(2)	9.81(6)	13.59(7)	7.69(5)	**0.32(1)**	5.34(4)
Reuters(21)	8.29(3)	9.64(5)	8.70(4)	9.80(6)	11.69(7)	**0.40(1)**	7.75(2)
ohsumed	344.83(6)	367.05(7)	142.82(2)	183.99(3)	284.95(4)	**11.44(1)**	326.35(5)
tmc2007	203.38(6)	195.54(5)	101.33(2)	107.86(3)	174.64(4)	**12.45(1)**	259.48(7)
Reuters(90)	13.43(6)	12.16(5)	6.66(2)	9.79(3)	22.93(7)	**1.99(1)**	10.71(4)
corel5k	22.11(5)	19.38(4)	**4.95(1)**	10.57(3)	30.92(6)	5.40(2)	51.99(7)
bibtex	308.37(6)	337.83(7)	40.76(2)	44.43(3)	112.75(4)	**3.56(1)**	220.54(5)
Avg. rank	5.182	4.727	2.636	3.545	5.727	**1.273**	4.909

performances on 11 data sets. We can see from Fig. 2 that the best performing methods for each performance measure are ML-Tree and HOMER, followed by CC and BR. Both ML-Tree and HOMER construct a hierarchy of tree derived from training classifiers at each node. The results of the experimental comparison revealed that a hierarchy of classifiers improves the prediction performance against a single BR classifier with independent classifiers.

We find that ML-Tree performs the best in terms of precision. On the other hand, HOMER performs the best in terms of recall. Precision and recall are two different quantitative measures. Precision is the fraction of predicted labels 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-Tree are more accurate than those from HOMER, while the predictions from HOMER are more complete than the ones from ML-Tree. A reasonable explanation for this finding is that, HOMER uses multiple-leaf labeling method to classify a new instance x. The union of the predicted labels in multiple leaves is used for classification purpose, and therefore the predicted labels are more complete. The ML-Tree uses decision-path labeling method that combining the predictive labels from the root to a leaf to make multilabel prediction. The prediction is obtained by a threshold function that is optimally tuned on the training set and therefore, is more accurate.

A closer examination of the results in Fig. 2 shows that CC and BR have average performance on the example-based evaluation measures. BR outperforms CC only on the hamming loss. The situation is reversed on the other example-based measures where CC and BR rank third and fourth, respectively, when evaluated by accuracy, precision, subset accuracy, and F1 score. This result indicates that CC and BR have similar performances, but the CC method is generally better.

Even though the Friedman test suggests that there are significant differences between the methods, most of the pairwise comparisons remain statistically nonsignificant (at a significance level of 5%). Nevertheless, the overall picture taken from the experiments is clearly in favor of ML-Tree and HOMER. The performances of ML-Tree and HOMER are often significantly better than those of PCT, ML-kNN, and IBLR-ML. ML-kNN and PCT perform poorly when evaluated by all evaluation measures and the later is the worst method among the seven multilabel learning algorithms. PCT is not competitive because the algorithm is not originally proposed for multilabel problems. It is not reliable for multilabel learning due to the inadequacy of modeling label correlations.

We further analyze the performance of the ML-Tree with CC and BR method. We can see that ML-Tree is better than CC and BR with respect to accuracy, precision, recall, F1 score, and subset accuracy. On the other hand, BR performs best in terms of hamming loss. Recent studies [38] show that BR is well tailored for hamming loss. The results in our experiments are in concordance with these studies.

In summary, the results of the experimental comparison illustrate the competitiveness of the ML-Tree method against

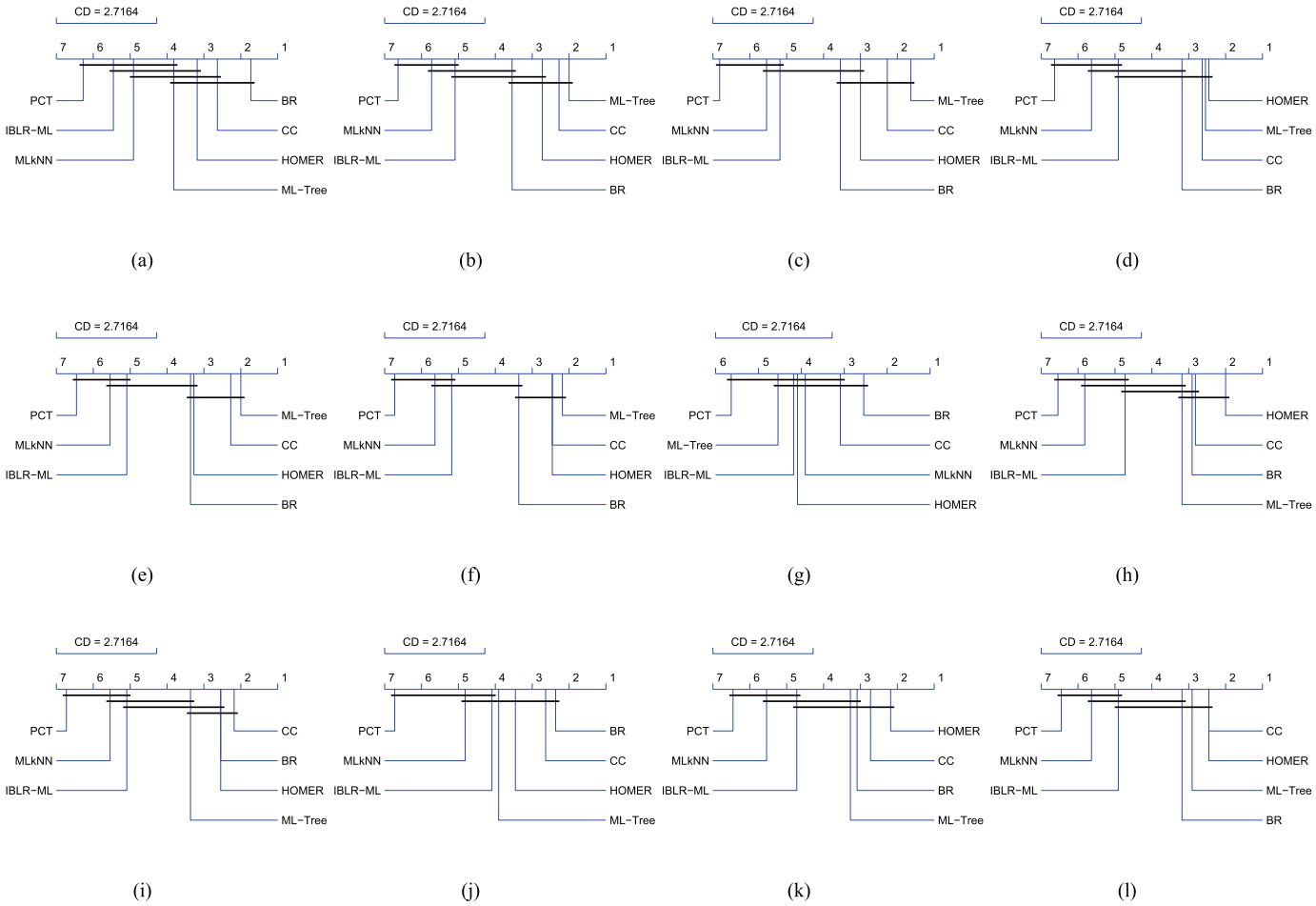

Fig. 2. Graphical presentation of results from the Neminyi post hoc test at 0.05 significance level for the example-based measures. (a) Hamming loss. (b) Accuracy. (c) Precision. (d) Recall. (e) Subset accuracy. (f) F1 score. (g) Microprecision. (h) Microrecall. (i) MicroF1. (j) Macroprecision. (k) Macrorecall. (l) MacroF1.

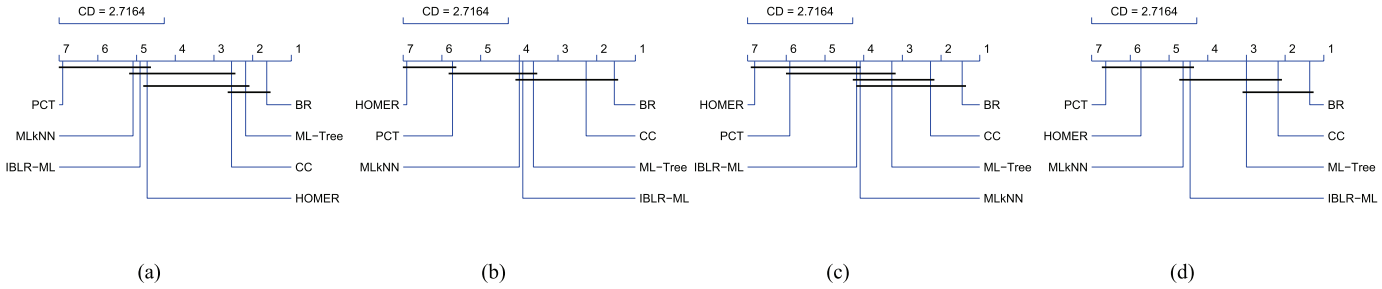

Fig. 3. Graphical presentation of results from the Neminyi post hoc test at 0.05 significance level for the ranking-based measures. (a) One-error. (b) Coverage. (c) Ranking-loss. (d) Average precision.

other multilabel learning methods. ML-Tree improves the predictive performances of the conventional BR method and PCT decision tree method.

E. Results on the Label-Based Measures

The label-based measures include microprecision, microrecall, microF1, macroprecision, macrorecall, and macroF1. The results for the statistical significance test are given in Fig. 2, while the complete results are shown in Tables XI–XVI. For the microF1 results, the best performing methods are CC, BR, HOMER, and ML-Tree. For the macroF1 results, the best performing methods are CC, HOMER, ML-Tree, and BR. We further compare BR with HOMER using the label-based measures. BR performs better when evaluated by precision (microprecision and macroprecision), while HOMER performs better with respect to recall (microrecall and macrorecall). The results indicate that HOMER is more complete, and BR is more accurate. The predicted labels by HOMER cover more ground-truth labels. On the other hand, the labels predicted by BR are more likely to be included in the ground-truth labels.

TABLE XXIII
TRAINING TIME AND HAMMING LOSS OF ML-TREE AND ML-TREE+ ON MULTILABEL DATA SETS

Dataset	Training Time		Hamming Loss	
	ML-Tree	ML-Tree+	ML-Tree	ML-Tree+
emotions	20 175	473	0.2343	0.2343
scene	33 500	4 068	0.0991	0.0991
yeast	63 297	18 009	0.0175	0.2426
medical	1 887	360	0.0134	0.015
Reuters(10)	44 242	4 148	0.0175	0.018
Reuters(21)	64 235	7 275	0.0127	0.0129
ohsumed	719 812	70 773	0.0522	0.0522
Reuters(90)	53 362	9 556	0.0057	0.0058
corel5k	308 374	132 117	0.01	0.01
bibtex	642 109	126 163	0.017	0.017

TABLE XXIV
COMPLEXITY AND PERFORMANCE OF ML-TREE IN TERMS OF AND HAMMING LOSS (HAM.) ON THE *scene* DATA SET

λ	#d.	#a.d.	#a.b.	#i.n.	#t.n.	#tr.	#te.	ham.
0.6	6	3.7	3.4	14	34	190	16	0.100
0.65	8	4.1	2.9	17	34	263	15	0.099
0.7	6	3.9	2.8	18	34	289	23	0.107
0.75	7	3.8	3.2	13	29	272	21	0.106
0.8	5	3.6	3.1	14	31	262	20	0.114
0.85	4	3.3	3.2	11	25	235	20	0.124
0.9	5	3.6	3.2	13	29	161	21	0.123
0.95	7	3.6	3.1	11	24	185	22	0.124

F. Results on the Ranking-Based Measures and Time Efficiency Measures

The ranking-based measures are one-error, coverage, ranking loss, and average precision. The results are given in Fig. 3 and Tables XVII–XX. The best performing method is BR, followed by CC and ML-Tree. ML-Tree is the second best in terms of one-error and the third best in terms of the other two measures (ranking-loss and average precision). Even though the HOMER approach is able to produce good results for the example-based measures, it performs poorly across all ranking-based evaluation measures. This result makes sense as it is shown in recent studies [34] that even good approaches are not able to outperform all other methods in every measure. The evaluation measures used in the experiments assess the learning performance from different aspects and one algorithm rarely outperforms another algorithm on all criteria, for example, a method that is optimal for instance hamming loss usually does not perform well in subset accuracy or one-error.

We also report the computation times (training time and testing time) of each comparing algorithm. The results are given in Tables XXI and XXII. The results show that PCT is the most efficient method, followed by ML-*k*NN and IBLR-ML. ML-Tree has the largest running time in the training phrase mainly due to the high computational cost in learning a hierarchy of classifiers. It is interesting to compare the running time of this method with the BR method as the tree model built in the root of the tree for the ML-Tree method is exactly the same as BR model. We see that the computational complexity of the ML-Tree could be comparable to those of the BR method for data sets with small label cardinality (e.g., the emotions and scene data sets). However, in the case of larger data sets with higher label cardinality, BR is much faster than ML-Tree.

TABLE XXV
SAME AS TABLE XXIV, BUT FOR THE *tmc2007* DATA SET

λ	#d.	#a.d.	#a.b.	#i.n.	#t.n.	#tr.	#te.	ham.
0.6	17	5.8	3.0	768	1541	3258	10	0.051
0.65	17	5.8	3.0	768	1541	3435	10	0.051
0.7	16	5.8	2.9	653	1269	3620	11	0.054
0.75	19	6.1	2.9	694	1328	3317	9	0.058
0.8	17	6.0	3.0	620	1213	3340	10	0.063
0.85	13	5.7	3.0	522	1069	3063	10	0.068
0.9	17	6.3	2.9	567	1089	3323	9	0.075
0.95	19	6.1	3.0	486	968	4184	11	0.087

TABLE XXVI
SAME AS TABLE XXIV, BUT FOR THE *corel5k* DATA SET

λ	#d.	#a.d.	#a.b.	#i.n.	#t.n.	#tr.	#te.	ham.
0.6	25	6.3	2.9	1143	2127	10254	327	0.0143
0.65	36	7.4	2.8	1080	1997	10254	327	0.0103
0.7	31	7.0	2.9	1088	2032	11936	332	0.0100
0.75	26	7.0	2.8	1123	2052	13584	335	0.0102
0.8	27	6.9	2.9	1044	2002	16322	330	0.0099
0.85	26	6.8	3.0	975	1910	10839	317	0.0100
0.9	31	7.3	2.9	966	1873	11727	369	0.0102
0.95	39	7.9	3.0	1001	1954	12640	347	0.0103

ML-Tree has two parameters to tune in the model: the SVM parameter C and the threshold parameter λ. Optimum selection of these two parameters increases the computational cost of the method. In an attempt to reduce the computational complexity of ML-Tree, we consider a simplified implementation of the ML-Tree method. For the simplified ML-Tree, called ML-Tree+, we set the parameter C simply to $C = 2^{-5}$ as the default, and optimally tune the threshold parameter λ in building the tree model. We perform experiments to compare the performance of the two implementations of our proposed method (ML-Tree and ML-Tree+). Table XXIII provides the hamming loss and training time (in seconds) results for the two methods. Table XXIII shows that ML-Tree+ is computationally more efficient than ML-Tree. In additional, ML-Tree+ is able to deliver comparable performance against ML-Tree. The performance of the SVM classifier with the default setting for parameter C learnt at each node of the ML-Tree+ may not be optimal, but growing a hierarchical of nodes and tuning an appropriate value for λ ensures good performance. The implementation of ML-Tree under optimal tuning C and λ has an advantage in classification performance. However, searching best value for the SVM classifiers at the nodes requires additional cost, and the computational complexity scales linearly with respect to the number of class labels and the number of nodes in the tree. When handling multilabel data sets with large label cardinality, the computations would become expensive. The choice of an appropriate implementation of ML-Tree essentially depends on the tradeoff between computational requirements and accuracy.

G. Parameter Study

Our experiment indicates that how different values of the parameter λ affect the construction time and the geometry of the resulting tree (maximum/average depth of the tree, the number of internal and leaf nodes, average branch-out) in the proposed method. We analyze the results of three data sets with small, medium, and large label cardinality l_c. We expect

TABLE XXVII
LABEL RELATIONSHIP ON 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	-0.18	-0.18	-0.10
l_2	1.00	1.00	-0.18	-0.18	-0.10
l_3	-0.18	-0.18	1.00	1.00	-0.14
l_4	-0.18	-0.18	1.00	1.00	-0.14
l_5	-0.10	-0.10	-0.14	-0.14	1.00

TABLE XXVIII
LABEL RELATIONSHIP ON 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	0.30	0.36	0.40	-0.06
l_2	0.30	1.00	0.33	0.25	-0.07
l_3	0.36	0.33	1.00	0.19	-0.06
l_4	0.40	0.25	0.19	1.00	-0.06
l_5	-0.06	-0.07	-0.06	-0.06	1.00

that the greater this statistic, the higher the complexity of the proposed method. The three data sets are scene ($l_c = 1.07$), tmc2007 ($l_c = 2.16$), and corel5k ($l_c = 3.52$). The results for the constructed trees with different parameter λ settings for different data sets are given in Tables XXIV–XXVI.

As shown in the tables, the best λ values for scene, tmc2007, and corel5k are 0.65, 0.60, and 0.80, respectively. The best parameter value is dependent on the evaluated data set. The value of the parameter λ should be properly chosen by cross validation on the training set. In our experiments, we tune the value of parameter λ from 0.5 to 1 using cross-validation approach on the training set. We find that the label cardinality l_c is an important factor affecting the complexity of the resulting tree. As expected, the average depth, the number of nodes, and the time of constructing a tree increase with l_c. Thus, we expect that the greater the label cardinality, the higher the complexity of the constructed tree.

H. Label Relationship

We use co-occurrence of predictive labels at the leaf nodes to estimate the label relationships. We construct a m-by-q matrix $\mathbf{Q}$ where m is the number of leaf nodes and q is the number of possible labels. The ith row of $\mathbf{Q}$ is the predictive label indicator vector $\mathbf{b}$ attached to the i-th leaf node. The entries taking the values of 1 correspond to the predictive labels in the leaf node. The ith column of $\mathbf{Q}$ is the distribution of label l_i extracted from the leaf nodes of the tree. The label relationship of two labels l_i and l_j can be measured using the ϕ-coefficient defined as follows:

$$\phi(i, j) = (AD - BC)/\sqrt{(A + B)(C + D)(A + C)(B + D)} \tag{3}$$

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. Our experiment examines whether our proposed algorithm can generate reasonable label relationships. As we have not given ground-truth label relationship for real-world data so far, we study two synthetic data sets [39] that we know the exact label relationship. In the experiments, we use them to check the validity of our proposed method for capturing the label relationships. The data sets have 10 000 instances and five labels 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 set $l_1 = l_2$ and $l_3 = l_4$. In the second data set $l_1 = l_2 \vee l_3 \vee l_4$. The label relationships for these two data sets are given in Tables XXVII and XXVIII, respectively. The diagonals are 1.0. 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 XXVII shows that the entries $(1, 2)$, $(2, 1)$, $(3, 4)$ and $(4, 3)$ are 1.0, while Table XXVIII shows that the entries $(1, 2)$, $(1, 3)$, $(1, 4)$ have relatively large positive values. These results are consistent with the ground-truth label relationship.

TABLE XXIX
EXPERIMENTAL RESULTS ON NUS-WIDE DATA SET IN TERMS OF DIFFERENT EVALUATION MEASURES

	BR	CC	HOMER	ML-Tree
Example-based measures:				
Hamming loss ↓	**0.0837**	0.0869	0.0868	0.0916
Accuracy ↑	0.3335	0.3163	**0.3610**	0.3421
Precision ↑	0.7190	0.7236	0.6554	**0.7329**
Recall ↑	0.3311	0.3614	**0.4170**	0.3359
F1 score ↑	0.4362	0.4687	**0.4994**	0.4510
Subset accuracy ↑	0.0012	0.0025	**0.0049**	0.0014
Label-based measures:				
Micro-precision ↑	0.7351	0.7186	0.6592	**0.7422**
Micro-recall ↑	0.3594	0.3297	**0.4166**	0.3236
Micro-F1 ↑	0.4827	0.4520	**0.5105**	0.4580
Macro-precision ↑	0.0839	0.1187	0.1268	**0.1356**
Macro-recall ↑	0.0575	0.0552	**0.0778**	0.0605
Macro-F1 ↑	0.0539	0.0536	**0.0703**	0.0657
Ranking-based measures:				
OneError ↓	0.1924	0.1822	0.1813	**0.1809**
Coverage ↓	20.210	20.920	35.880	**19.550**
Ranking loss ↓	0.1612	0.1721	0.3784	**0.1553**
Average precision ↑	0.6064	0.5748	0.4815	**0.6139**

I. Large-Scale Multilabel Classification of Flickr Images

In this experiment, we test the performance of ML-Tree on a relatively large-scale multilabel data set with large label cardinality. We use a real-world Web image data set from the NUS-WIDE data set.[3] The data set has 81 ground-truth concepts that can be used for multilabel evaluation. These concepts include general concepts such as animal and specific concepts such as dog and cat. The data set includes 269 648 Flickr images which are separated into 161 789 training images and 107 859 testing images. As we consider large label cardinality, we remove images with less than two class labels and ground-truth concepts occurring in less than 1% images of the data set. The remaining set contains 26 516 training images, 17 906 test images and 44 ground-truth concepts. The label cardinality of this data set is about 5.0.

We compare the performance of ML-Tree algorithm with HOMER, BR, and CC. We do not empirically tune the parameter C in the SVMs of BR, CC, HOMER, and

[3] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

ML-Tree. We simply use the default value $C = 2^{-5}$. The experimental results evaluated using the 16 different measures are summarized in Table XXIX. The results are grouped into three categories: example-based, label-based, and ranking-based. We can see that the results are in line with our previous observations: the best performing methods are ML-Tree and HOMER, ML-Tree performs the best in terms of precision, while ML-Tree performs the best in terms of recall. Interestingly, ML-Tree shows consistently better performances on ranking-based evaluation measures. The ranking-based measures include one-error, ranking loss, coverage, and average precision. These measures offer a different perspective on the results. The classifiers are evaluated in terms of its ability to provide a good approximation of ranks for labels associated with an unlabeled instance. The experimental results indicate that ML-Tree can generate a ranking of possible labels for a given instance such that its correct labels receive higher ranking than irrelevant labels. The HOMER method only considers the predictions on the leaves and uses relevant information that comes from the outputs of the classifiers just above the corresponding leaves. On the other hand, our proposed ML-Tree approach shares the relevant information across the tree model, where the ranking scores of the classes are obtained by summarizing the relevance scores from the root to the leaf node. The relevant information of the general concepts from parent nodes propagates to the child nodes that have specific concepts. Therefore, it may be counted multiple times to ensure a higher chance of achieving a high rank result. Such a relevant scoring method imposes a well impact on ranking-based evaluation performance for the Flickr multilabel image classification data set that contains general concepts and specific concepts.

V. Conclusion

In this paper, we present a new hierarchical tree approach (ML-Tree) for multilabel learning. The ML-Tree uses one-against-all SVM classifiers and the transmission of predictive labels at each node to grow up a hierarchical tree model. It provides a new multilabel classification framework, where multiple one-against-all SVM classifiers at different levels of a tree can effectively work together to reveal the correlations among labels and to predict the labels of an instance more precisely. We examine the ML-Tree method on 11 real data sets from different domains and compare it with BR, CC, ML-kNN, IBLR-ML, HOMER, and PCT. We employ Friedman and Nemenyi tests to assess the statistical significance of the differences in performance. The experimental results demonstrate the effectiveness of our proposed ML-Tree method. Note that ML-Tree has the largest running time in the training phase, which is mainly due to the high computational cost in learning a hierarchy of classifiers. In the future work, we will consider more efficient approaches to construct the tree to reduce the complexity of the algorithm. We also plan to investigate the sensitivity of the hierarchical tree model to outliers in the training data and to extend the ML-Tree method to an ensemble framework. Moreover, it will be interesting to investigate the theoretical aspects of the proposed method and the influence of the relations among labels on multilabel learning.

References

[1] R. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, no. 2, pp. 135–168, 2000.

[2] F. De Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *Proc. 3rd Int. Conf. MLDM Pattern Recognit.*, 2003, pp. 251–274.

[3] H. Kazawa, T. Izumitani, H. Taira, and E. Maeda, "Maximal margin labeling for multi-topic text categorization," in *Proc. Adv. NIPS*, vol. 17. 2005, pp. 649–656.

[4] N. Ueda and K. Saito, "Parametric mixture models for multi-labeled text," in *Proc. Adv. NIPS*, vol. 15. 2003, pp. 721–728.

[5] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Proc. Adv. NIPS*, vol. 14. 2002, pp. 681–687.

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

[7] A. Clare and R. King, "Knowledge discovery in multi-label phenotype data," in *Proc. Principles Data Mining Knowl. Discovery*, 2001, pp. 42–53.

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

[9] M. Zhang and Z. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, 2007.

[10] 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.

[11] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th Int. Conf. Music Inf. Retrieval*, 2008, pp. 325–330.

[12] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int. J. Data Warehousing, Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[13] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer-Verlag, 2010, pp. 667–685.

[14] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, May 2013, to be published.

[15] 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. Retrieval*, 2005, pp. 274–281.

[16] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proc. 14th ACM Int. Conf. Inf. Knowl. Manag.*, 2005, pp. 195–200.

[17] K. Gold and A. Petrosino, "Using information gain to build meaningful decision forests for multilabel classification," in *Proc. IEEE 9th Int. Conf. Develop. Learn.*, Jan. 2010, pp. 58–63.

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

[19] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proc. 15th Int. Conf. Mach. Learn.*, 2000, pp. 55–63.

[20] 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.

[21] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of multi-objective decision trees," in *Proc. ECML*, 2007, pp. 624–631.

[22] I. W. Tsang, J. T. Kwok, and P. M. Cheung, "Core vector machines: Fast SVM training on very large datasets," *J. Mach. Learn. Res.*, vol. 6, no. 1, pp. 363–392, 2006.

[23] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD Workshop Mining Multidimensional Data*, 2008, pp. 30–44.

[24] 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.

[25] K. Punera, S. Rajan, and J. Ghosh, "Automatically learning document taxonomies for hierarchical classification," in *Proc. Special Interest Tracks Posters 14th Int. Conf. World Wide Web*, 2005, pp. 1010–1011.

[26] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei, "Fast and balanced: Efficient label tree learning for large scale object recognition," *Adv. Neural Inf. Process. Syst.*, vol. 2, pp. 567–575, Dec. 2011.

[27] 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.

[28] B. Fu, Z. Wang, R. Pan, G. Xu, and P. Dolog, "Learning tree structure of label dependency for multi-label learning," in *Proc. Adv. Knowl. Discovery Data Mining*, 2012, pp. 159–170.

[29] 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.

[30] N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *Adv. Neural Inf. Process. Syst.*, vol. 7, no. 1, pp. 233–240, 2005.

[31] W. Bi and J. T. Kwok, "Multi-label classification on tree- and DAG-structured hierarchies," in *Proc. 28th ICML*, 2011, pp. 17–24.

[32] M. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 999–1007.

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

[34] 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.

[35] 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, Jan. 2010.

[36] C. Chang and C. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.

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

[38] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "Regret analysis for performance metrics in multi-label classification: The case of Hamming and subset zero-one loss," in *Proc. Mach. Learn. Knowl. Discovery Databases*, 2010, pp. 280–295.

[39] S. Huang, Y. Yu, and Z. Zhou, "Multi-label hypothesis reuse," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 525–533.

Qingyao Wu (S'11) received the B.Sc. degree from the South China University of Technology, Guangzhou, China, and the M.Sc. and Ph.D. degrees in computer science from the Shenzhen Graduate School, Harbin Institute of Technology, Harbin, China, in 2007, 2009, and 2013, respectively.

He is currently a Post-Doctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include machine learning, data mining, and bioinformatics.

Yunming Ye (M'04) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China.

He is currently a Professor with the Shenzhen Graduate School, Harbin Institute of Technology, Harbin, China. His current research interests include data mining, text mining, and ensemble learning algorithms.

Haijun Zhang received the B.Eng. and master's degrees from Northeastern University, Shenyang, China, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2004, 2007, and 2010, respectively.

He was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, from 2010 to 2011. Since 2012, he has been with the Shenzhen Graduate School, Harbin Institute of Technology, Harbin, China, where he is currently an Associate Professor of Computer Science. His current research interests include multimedia data mining, machine learning, pattern recognition, evolutionary computing, and communication networks.

Tommy W. S. Chow (M'93–SM'03) received the B.Sc. (Hons.) and Ph.D. degrees from the University of Sunderland, Sunderland, U.K.

He joined the City University of Hong Kong, Hong Kong, as a Lecturer, in 1988, where he is currently a Professor with the Electronic Engineering Department. He was with NEI Reyrolle Technology, Hebburn, U.K., where he was developing digital simulator for transient network analyzer. He was involved in a research project involving high-current density current collection system for superconducting direct current machines, in collaboration with the Ministry of Defense (Navy), Bath, U.K., and the International Research and Development, Newcastle upon Tyne, U.K. He has authored and co-authored more than 170 technical papers in international journals, five book chapters, and more than 60 technical papers in international conference proceedings. His current research interests include machine learning, including supervised and unsupervised learning, data mining, pattern recognition, and fault diagnostic.

Shen-Shyang Ho (M'07) received the B.S. degree in mathematics and computational science from the National University of Singapore, Singapore, and the M.S. and Ph.D. degrees in computer science from George Mason University, Fairfax, VA, USA, in 1999, 2003, and 2007, respectively.

He was a NASA Post-Doctoral Program Fellow and then a Post-Doctoral Scholar with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, from 2007 to 2010. From 2010 to 2012, he was a Researcher with the University of Maryland Institute for Advanced Computer Studies, College Park, MD, USA, where he was involved in projects funded by NASA. He is currently a Tenure-Track Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. He is currently involved in the industrial research projects funded by BMW and Rolls-Royces. He has given tutorials at the Association for the Advancement of Artificial Intelligence, the International Joint Conference on Neural Networks, and the European Conference on Machine Learning. His current research interests include data mining, machine learning, pattern recognition in spatiotemporal/data streaming settings, and privacy issues in data mining.