

Multi-label Classification without the Multi-label Cost

Xiatian Zhang* Quan Yuan* Shiwan Zhao* Wei Fan† Wentao Zheng* Zhong Wang‡

Abstract

Multi-label classification, or the same example can belong to more than one class label, happens in many applications. To name a few, image and video annotation, functional genomics, social network annotation and text categorization are some typical applications. Existing methods have limited performance in both efficiency and accuracy. In this paper, we propose an extension over decision tree ensembles that can handle both challenges. We formally analyze the learning risk of Random Decision Tree (RDT) and derive that the upper bound of risk is stable and lower bound decreases as the number of trees increases. Importantly, we demonstrate that the training complexity is independent from the number of class labels, a significant overhead for many state-of-the-art multi-label methods. This is particularly important for problems with large number of multi-class labels. Based on these characteristics, we adopt and improve RDT for multi-label classification. Experiment results have demonstrated that the computation time of the proposed approaches is 1-3 orders of magnitude less than other methods when handling datasets with large number of instances and labels, as well as improvement up to more than 10% in accuracy as compared to a number of state-of-the-art methods in some datasets for multi-label learning. Considering efficiency and effectiveness together, Multi-label RDT is the top rank algorithm in this domain. Even compared with the HOMER algorithm proposed to solve the problem of large number of labels, Multi-label RDT runs 2-3 orders of magnitude faster in training process and achieves some improvement on accuracy. Software and datasets are available from the authors.

1 Introduction

In recent years, there has been much study in multi-label classification problem as motivated from emerging applications. Tsoumakas et al. [10] discussed its

applications in image, video, text annotation and categorization, gene function classification, and so on. In multi-label classification, an example is associated with a set of labels Y , where $Y \subseteq L$, L is the set of all labels, $|L| \geq 3$ and $|Y| \geq 2$. For example, a news article about global warming can be classified into 3 categories, climatology, policy, and society at the same time.

Existing methods for multi-label classification fall into two main categories [10]: problem transformation and algorithm adaptation. Problem transformation maps the multi-label learning problem into one or more single-label problems. Label Powerset (LP) and Binary Relevance (BR) are two problem transformation methods. LP takes each unique combination of original class labels as a new label resulting in up to $2^{|L|}$ transformed labels. On the other hand, BR trains a single-label classifier for each label. Algorithm adaptation extends specific learning algorithms in order to handle multi-label data directly, and examples include SVM, decision tree, neural network, lazy learning, Bayesian, boosting, etc. The multiple labels and large size of label combinations make multi-label classifiers 1 or 2 magnitudes slower than single-label classification on the same number of examples and features. Application with large number of class labels (e.g., 500) [12] can virtually “kill” most of these methods. HOMER algorithm [12] was proposed to solve the problem, but the training complexity of HOMER still depends on $|L|$. In addition to the efficiency problem, the capabilities of many existing methods partly depend on the selected single-label classifier, because all problem transformation methods and some algorithm adaptation methods are meta algorithms. For example, the popular method SVM needs to be hand tuned to support nominal attributes and nonlinear datasets.

In this paper, we propose to adopt Random Decision Tree (RDT) [27] for multi-label classification to overcome the limitations mentioned above by LP and BR methods. First, we formally analyze the learning risk of Random Decision Tree and derive that the upper bound of risk is stable and lower bound decreases as the number of trees increases, which guarantees high accuracy and robust performance on various datasets. Second, through computation complexity analysis, we show that the time complexities of training process of both

*IBM Research - China. {xiatianz, quanyuan, zhaosw, zhengwt}@cn.ibm.com.

†IBM T. J. Watson Research Center, USA. weifan@us.ibm.com.

‡Northeast University, Shenyang, China. wangzhong.neu@gmail.com.

LP-RDT and BR-RDT are $O(mn \log(n))$, m is the number of trees, n is the number of instances and $m \ll n$. Therefore the training computation cost of Multi-label RDT (ML-RDT) is low and independent of label size. In practice, this means both LP-RDT and BR-RDT can handle problems with large number of labels efficiently without paying the multi-label cost. Third, besides the two main advantages, another strength of RDT is that it is based on decision tree so that it can handle nominal attributes, missing values, and nonlinear classification problems without modification.

We compare the effectiveness and efficiency of LP-RDT and BR-RDT approaches with KNN, C4.5, Naive Bayes, SMO [15] on five small datasets from different application domains. Experiment results also showed that ML-RDT (Multi-label RDT) methods can improve up to 10% in accuracy compared to state-of-the-art approaches on a number of datasets. Multi-label RDT outperforms other methods on two out of five small datasets and keeps close to top one on other three datasets. ML-RDT has robust and reliable performance across different datasets. Experiment results also demonstrate that the computation time of the proposed approach is 1-3 orders of magnitude less than other methods on the datasets with 500 labels or more than 48,000 instances. Moreover, we compare BR-RDT with HOMER on the delicious dataset, which contains 983 labels and more than 10 thousand instances. HOMER [12] is an algorithm proposed for the problem of large number of labels. The comparison results show BR-RDT is faster and better than HOMER, especially in training process, where BR-RDT runs about 2-3 orders of magnitude faster than HOMER.

Table 1: Symbols

D	dataset of instances
T	Training dataset
X	Instance space
x	Instance
L	Label set
λ	Label
Y	Vector of real labels
y	Real label
Z	Vector of predicted labels
z	Predicted label
H	Set of classifiers
h	Classifier
ε	Real error rate of a classifier
$\hat{\varepsilon}$	Empirical error rate of a classifier
c	Confidence of a class

2 Multi-label Classification via RDT

We propose a straightforward extension to use random decision trees for efficient multi-label classification. We adopt both Label Powerset (or LP) and Binary Relevance (or BR), to transform a multi-label problem to multiple single-label problems.

2.1 Random Decision Trees Random decision tree is proposed by Fan et al. [27]. The main difference between RDT and other classical decision tree methods, such as C4.5, is that RDT constructs decision tree randomly and doesn't use any information of labels. That nature makes RDT fast and the computational cost independent of labels. In the rest of this subsection, we will briefly introduce how RDT works.

Random decision tree algorithm constructs multiple decision trees randomly. When constructing each tree, the algorithm picks a "remaining" feature randomly at each node expansion without any purity function check (such as information gain, gini index, etc.). A categorical feature (such as gender) is considered "remaining" if the same categorical feature has not been chosen previously in a particular decision path starting from the root of tree to the current node. Once a categorical feature is chosen, it is useless to pick it again on the same decision path because every example in the same path will have the same value (either male or female). However, a continuous feature (such as income) can be chosen more than once in the same decision path. Each time the continuous feature is chosen, a random threshold is selected.

A tree stops growing any deeper if one of the following conditions is met:

1. The number of examples on a node is less than or equal to the assigned number.
2. The depth of tree exceeds some limits.

Each node of the tree records class distributions. Assume that a node has a total of 1000 examples from the training data that pass through it. Among these 1000 examples, 200 of them are + and 800 of them are -. Then the class probability distribution for + is $P(+|x) = 200/1000 = 0.2$ and the class probability distribution for - is $P(-|x) = 800/1000 = 0.8$.

The algorithm does not prune the randomly built decision tree in the conventional sense (such as MDL-based pruning and cost-based pruning, and etc.). However, it does remove "unnecessary" nodes. A node expansion is considered unnecessary, if none of its descendants have significantly different class distribution from this node. When this happens, we remove the expansion from this node and make the node a leaf node. In

our implementation, the random tree is built recursively and “necessity check” is performed when the recursion returns.

Classification is always done at the leaf node level. Each tree outputs a class probability distribution (as defined above). The class distribution outputs from multiple trees are “simply” averaged as the final class distribution output. For example, assume that there are two trees. Tree 1 outputs $P(+|x) = 0.2$ and $P(-|x) = 0.8$, and tree 2 outputs $P(+|x) = 0.3$ and $P(-|x) = 0.7$. The averaged probability output will be $P(+|x) = (0.2 + 0.3)/2 = 0.25$, and $P(-|x) = (0.8 + 0.7)/2 = 0.75$.

In some situations, a leaf node could be empty. When this happens, the algorithm doesn’t output NaN (not a number) or 0 as the class probability distribution, but it goes one level up to its parent node and output its parent’s class distribution.

In order to make a final prediction, a loss function is needed. For example, under traditional 0-1 loss, the best prediction is to choose the class label that is most likely. For example, for the binary problem, such as + and -, we predict class label + iff $P(+|x) \geq 0.5$. In some situations, when the training data and testing data are not drawn with the same probability, the optimal decision threshold may not be exactly 0.5.

2.2 LP-RDT Label Powerset considers each unique subset of labels that exists in the multi-label dataset as a single label. Let L be the set of all labels, $L = \{l_1, l_2, \dots, l_{|L|}\}$. $P(L)$ is the power set of L , and $|P(L)| = 2^{|L|}$. Each element in $P(L)$ stands for one possible combination of labels. For example, if there are 3 labels, and for each label, it will appear (+) or not (-), so the number of power set $P(L)$ is 8, including combinations like (+++), (-++), (+-+), etc. Each combination is considered as one class of traditional single-label classification, thus we can consider (+++) as a new label A; (-++) as B, etc. So the n -label classification problem is transformed to a single-label problem with at most $2^{|L|}$ class labels. After the transformation, it becomes a pure single-label multi-class classification problem. LP method introduces a large number of classes. For traditional decision tree algorithm, the large number of classes causes much computation cost when computing the splitting criterion on every inner node. During the procedure to construct random trees, except for a leaf node that summarizes class label distribution, there is no need to use the training examples’ class labels. Except on leaf node, the multi-class random tree building procedure is the same with binary classification tree. It is important to understand that the large number of classes has no effect on the computation cost

of random tree construction and this is an important property for problems with large number of multi-labels.

2.3 BR-RDT Binary Relevance (or BR) learns one binary classifier h_k for each label λ_k . It builds $|L|$ datasets by using all the instances with one label λ_k each time. To classify an unlabeled instance, BR generates the final multi-label prediction by summarizing the output of $|L|$ classifiers. The problem of BR method is that it needs as many classifiers as the number of labels. When the number of labels is large, the training and test computation cost becomes significant. As a random tree constructed is actually independent from class labels, it can instead classify all labels, and an ensemble of random trees can be used for the problem with one to even hundreds of labels. The added trivial computation is only the label probability distribution counting on each leaf node.

3 Learning Risk Analysis of RDT

In this section, we will analyze the learning risk of RDT. Through the analysis, we claim that learning risk of RDT is stable.

The risk metric of single-label RDT is the error rate:

$$\varepsilon(H, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \text{Sgn}(Z_i)$$

where,

$$\text{Sgn}(Z_i) = \begin{cases} 1 & Z_i \neq Y_i \\ 0 & Z_i = Y_i \end{cases}$$

For simplicity, we consider binary-classification problem with two classes + and -. At first, we study the training error of RDT. For an instance x in training dataset T and the k -th tree, the training error of the leaf node containing x is $\hat{\varepsilon}(h_k, x)$, then the training accuracy is $1 - \hat{\varepsilon}(h_k, x)$. In the following, we simplify $\hat{\varepsilon}(h_k, x)$ to $\hat{\varepsilon}_{k,x}$. Fan et al. [27] found that in any leaf node of a decision tree, $\hat{\varepsilon}_{k,x} \leq 1/2$. Let $c_{+,x}$ and $c_{-,x}$ be the average probabilities of x on all the trees for + and - respectively.

Given x in T is +. If h_k classifies it correctly as +, then h_k contributes $1 - \hat{\varepsilon}(h_k, x)$ to the $c_{+,x}$ and $\hat{\varepsilon}(h_k, x)$ to the $c_{-,x}$.

If $|H| = 2$, there are 4 possible classification combinations, ++, +-, -+, and --. The probabilities of all combinations are:

$$(1 - \hat{\varepsilon}_{1,x})(1 - \hat{\varepsilon}_{2,x}), (1 - \hat{\varepsilon}_{1,x})\hat{\varepsilon}_{2,x}, \hat{\varepsilon}_{1,x}(1 - \hat{\varepsilon}_{2,x}), \hat{\varepsilon}_{1,x}\hat{\varepsilon}_{2,x}$$

And the $c_{+,x}$ of all combinations are:

$$\begin{aligned} & ((1 - \hat{\varepsilon}_{1,x}) + (1 - \hat{\varepsilon}_{2,x}))/2, \quad ((1 - \hat{\varepsilon}_{1,x}) + \hat{\varepsilon}_{2,x})/2 \\ & (\hat{\varepsilon}_{1,x} + (1 - \hat{\varepsilon}_{2,x}))/2, \quad (\hat{\varepsilon}_{1,x} + \hat{\varepsilon}_{2,x})/2 \end{aligned}$$

Assume $0 \leq \hat{\varepsilon}_{1,x} \leq \hat{\varepsilon}_{2,x} \leq 1/2$, then the $c_{+,x}$ of ++ and +- are greater than 1/2, $c_{+,x}$ of -+ and -- are less than 1/2. Then,

$$\begin{aligned} \Pr\{c_{+,x} < 1/2\} &= \hat{\varepsilon}_{1,x}(1 - \hat{\varepsilon}_{2,x}) + \hat{\varepsilon}_{1,x}\hat{\varepsilon}_{2,x} \\ &= \hat{\varepsilon}_{1,x} \leq 1/2 \end{aligned}$$

If x in T is -, as the same way, there is also:

$$\begin{aligned} \Pr\{c_{-,x} < 1/2\} &= \hat{\varepsilon}_{1,x}(1 - \hat{\varepsilon}_{2,x}) + \hat{\varepsilon}_{1,x}\hat{\varepsilon}_{2,x} \\ &= \hat{\varepsilon}_{1,x} \leq 1/2 \end{aligned}$$

The above formulas say that for 2 trees, a given x in T will be classified incorrectly with the probability, $\min(\hat{\varepsilon}_{1,x}, \hat{\varepsilon}_{2,x})$. Apparently, the training error on the two-tree ensemble will be less than the error on each tree. Next we will prove that the same effect of two trees can be generalized to multiple trees.

THEOREM 3.1. *For the binary classification RDT. Given H , $\forall x$ in T , then,*

$$\Pr\{c_{z=y,x} < 1/2\} = \min(\hat{\varepsilon}_{1,x}, \hat{\varepsilon}_{2,x}, \dots, \hat{\varepsilon}_{|H|,x}) = \hat{\varepsilon}_{min,x}^H$$

Proof. If $|H| \neq 2^N$, $N \geq 1$. we can easily extend the size of H to 2^N by copying some trees in H . We just discuss it when $|H| = 2^N$. We use mathematical induction method:

1) According to the conclusion above, when $N = 1$, Above equation is true.

2) Assume when $N = q$, the theorem is true. While $N = q + 1$, we can split the H to two subsets H_1 and H_2 with the same size 2^q . We take the H_1 and H_2 as two classifiers. In addition, there are $\hat{\varepsilon}_{H_1,x} = \hat{\varepsilon}_{min,x}^{H_1}$, and $\hat{\varepsilon}_{H_2,x} = \hat{\varepsilon}_{min,x}^{H_2}$. So, $\hat{\varepsilon}_{H,x} = \hat{\varepsilon}_{min,x}^H$. That means when $N = q + 1$, the Theorem 4.1 established.

Based on 1) and 2), $\Pr\{c_{z=y,x} \leq 1/2\} = \hat{\varepsilon}_{min,x}^H$.

■

According to the result of Theorem 4.1, since the classified errors for training instances decrease monotonically with the increasing of the number of trees, the expected training risk of RDT will also decrease.

After deriving that RDT can minimize the training error, we analyze the generalization error of RDT. Given x in X is +. If h_k classifies it correctly as +, then h_k contributes $1 - \hat{\varepsilon}(h_k, x)$ to the $c_{+,x}$ and $\hat{\varepsilon}(h_k, x)$ to the $c_{-,x}$.

If $|H| = 2$, there are 4 possible classification combinations, ++, +-, -+, and --. The probabilities of all combinations are:

$$(1 - \varepsilon_{1,x})(1 - \varepsilon_{2,x}), (1 - \varepsilon_{1,x})\varepsilon_{2,x}, \varepsilon_{1,x}(1 - \varepsilon_{2,x}), \varepsilon_{1,x}\varepsilon_{2,x}$$

And the $c_{+,x}$ of all combinations are:

$$\begin{aligned} &((1 - \hat{\varepsilon}_{1,x}) + (1 - \hat{\varepsilon}_{2,x}))/2, \quad ((1 - \hat{\varepsilon}_{1,x}) + \hat{\varepsilon}_{2,x})/2 \\ &(\hat{\varepsilon}_{1,x} + (1 - \hat{\varepsilon}_{2,x}))/2, \quad (\hat{\varepsilon}_{1,x} + \hat{\varepsilon}_{2,x})/2 \end{aligned}$$

Assume $0 \leq \hat{\varepsilon}_{1,x} \leq \hat{\varepsilon}_{2,x} \leq 1/2$, then the $c_{+,x}$ of ++ and +- are greater than 1/2, $c_{+,x}$ of -+ and -- are less than 1/2.

Then,

$$\begin{aligned} \Pr\{c_{+,x} < 1/2\} &= \varepsilon_{1,x}(1 - \varepsilon_{2,x}) + \varepsilon_{1,x}\varepsilon_{2,x} \\ &= \varepsilon_{1,x} \end{aligned}$$

In this case, $\varepsilon_{1,x} \leq \varepsilon_{2,x}$ is not always established. However, since $\hat{\varepsilon}_{2,x} - \hat{\varepsilon}_{1,x} \geq 0$ and the probability density function of $\varepsilon_{2,x} - \varepsilon_{1,x}$ are symmetrical with $\hat{\varepsilon}_{2,x} - \hat{\varepsilon}_{1,x}$, $\Pr\{\varepsilon_{2,x} - \varepsilon_{1,x} \geq 0\} \geq 1/2$. The result is also true when x in T is -. When $\varepsilon_{1,x} \geq \varepsilon_{2,x}$, there is $\Pr\{\varepsilon_{1,x} - \varepsilon_{2,x} \geq 0\} \geq 1/2$. As the result, $\rho^H(x) = \Pr\{\varepsilon_x^H = \varepsilon_{min,x}^H\} \geq 1/2$, where ε_x^H is the error rate of x on the combined 2 trees. Then,

$$\begin{aligned} \varepsilon(H, X) &= \int_x^X \varepsilon_x^H dF(x) \\ &= \int_x^X \varepsilon_{min,x}^H \rho^H(x) dF(x) \\ &+ \int_x^X \varepsilon_{max,x}^H (1 - \rho^H(x)) \\ &\leq \frac{1}{2} \int_x^X (\varepsilon_{1,x} + \varepsilon_{2,x}) dF(x) \end{aligned}$$

, where $F(x)$ is the probability distribution function of x on X .

Above result can be easily extended to the situation $|H| > 2$. Like the proof of Theorem 4.1, we can prove:

THEOREM 3.2. *For the binary classification RDT. Given H , $\forall x$ in T , then,*

$$\varepsilon(H, X) \leq \frac{1}{|H|} \sum_{i=1}^{|H|} \varepsilon(h_i, X)$$

■

Theorem 4.2 says that the risk bound of RDT will not increase with the increasing of the number of trees and the risk doesn't exceed the average risk of all the trees. However, the Theorem 4.2 is a pessimistic result. In the optimistic situation, i.e., when $0 \leq \hat{\varepsilon}_{1,x} \leq \hat{\varepsilon}_{2,x} \leq 1/2$, there is always $0 \leq \varepsilon_{1,x} \leq \varepsilon_{2,x} \leq 1/2$. In this situation, as the same proof of Theorem 4.1, we can get,

$$(1 - \varepsilon_{1,x})(1 - \varepsilon_{2,x}), (1 - \varepsilon_{1,x})\varepsilon_{2,x}, \varepsilon_{1,x}(1 - \varepsilon_{2,x}), \varepsilon_{1,x}\varepsilon_{2,x} \quad \Pr\{c_{z=y,x} < 1/2\} = \min(\varepsilon_{1,x}, \varepsilon_{2,x}, \dots, \varepsilon_{|H|,x}) = \varepsilon_{min,x}^H$$

That says that under the optimistic situation, RDT can minimize the generalization error with the increasing of the number of trees.

The practical situation is always between the pessimistic and optimistic situations. That means,

$$\int_x^X \varepsilon_{min,x}^H dF(x) \leq \varepsilon(H, X) \leq \frac{1}{|H|} \sum_{i=1}^{|H|} \varepsilon(h_i, X)$$

Above inequations indicates the upper bound of learning risk is stable and the lower bound decreases with the increasing of the number of trees.

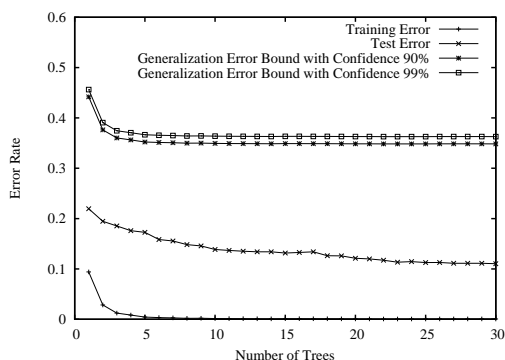


Figure 1: Errors change with the increasing of the number of trees on the scene dataset

In order to evaluate the theoretic result, we do experiments to investigate the variety of the training and test errors on the scene dataset (the average error of all the labels). The Figure 1 illustrates both the training and test errors decrease with the increasing of the number of trees. That means with the increasing of tree number, RDT reduces the training and test errors on the same time. The result fits the analysis above well.

4 Computation Complexity Analysis

Previous work [27, 25, 26] reported the low training cost of RDT for binary classification and regression by experimental results. We formally analyze the computation complexity of Multi-label RDT in this section. According to the analysis results, we can claim that both LP-RDT and BR-RDT are multi-label classifiers without the multi-label cost.

4.1 Training Complexity Generally speaking, ensemble learning methods take high computation cost for training many learners. However, [27] informally claimed that RDT has low computation complexity of

training, but it didn't give the concrete Big-O estimation.

The computation cost of a random tree consists of two parts. One is the tree construction cost, and the other is the distribution counting on the leaf nodes. Because the tree construction procedure is the same for single-label RDT, LP-RDT and BR-RDT, the computation complexity is also the same for the 3 variations of RDT. The computation complexity of tree construction depends on the numbers of instances and attributes of the training dataset. But in the worst case, it is just restricted by the number of instances. The worst case is that each leaf node only contains one training instance, and then the number of leaf nodes is n . Given the number of leaf nodes, the tree with the maximal nodes is the balanced binary tree, then the depth of the tree is $\lceil \log_2(n) \rceil$ excluding leaf nodes. During the constructing stage, each level should distribute all the n instances to the next level, so all the instances will be distributed $n \cdot \lceil \log_2(n) \rceil$ times. Then the time complexity of constructing a tree is $O(n \lceil \log_2(n) \rceil)$. The second part of training computation complexity is counting the probability distributions of the labels or classes on each leaf node. That part is a little different between LP-RDT and BR-RDT. Single-label RDT is the same with LP-RDT. For LP-RDT, each training instance will be counted one time. In this case, there are n additions for counting. Compared with the complexity of tree construction, the complexity of distribution counting can be ignored. Considering there are m trees, the training computation complexity of LP-RDT is $O(mn \log(n))$, where $m \ll n$. For BR-RDT, a training instance can take several labels. Given the average number of labels of training instances is t , there are tn additions for counting the probability distributions of labels. Therefore, the training computation complexity of BR-RDT is $O(m(n \log(n) + tn))$. In common case, $t \ll n$ and $t \ll |L|$. That means the effect of tn term is limited for the computation cost. According to the training complexities of both LP-RDT and BR-RDT, for general problems, the training computation costs of both LP-RDT and BR-RDT are irrelevant with the label size $|L|$.

The complexity of Top-Down Induction of Decision Trees (TDIDT), the family of algorithms typified by ID3 and C4.5, is $O(d^2n)$ [16], where $d = \sum (V_i - 1)$ and V_i is the size of values of i -th attribute. In common case, $m \log(n) \ll d^2$, so the complexity of RDT is much less than TDIDT. Apparently, for BR-C4.5 algorithm, the complexity becomes $O(|L|d^2n)$. For the large number of labels problems, the training computational cost becomes very huge.

The HOMER algorithm [12] was proposed to handle

Table 2: Statistics of the three datasets

Metric	Attributes							
	domain	#instances	#nominal	#numeric	#labels	cardinality	density	#distinct
genbase	biology	662	1186	0	27	1.252	0.046	32
yeast	biology	2417	0	103	14	4.237	0.303	198
scene	multimedia	2407	0	294	6	1.074	0.179	15
emotions	music	593	0	72	6	1.869	0.311	27
enron	text	1702	1001	0	53	3.378	0.064	753
delicious train/test	text	12920/3185	500	0	983	19.020	0.019	15806

the multi-label classification problems with large number of labels. When ignoring the numbers of instances and features, the training computation complexity of HOMER is $O(f(|L| + |L|))$, where $f(|L|)$ is the complexity of the balanced clustering algorithm with respect to a set of labels L . Obviously, HOMER's training cost increases more than linearly with the increasing of $|L|$.

4.2 Test Complexity The test computation complexity of RDT is like the test complexity of other decision trees, except RDT has m trees. Assume the average depth of the branches that test instance belong to is q , the decision tree's test computation complexity is $O(nq)$, where n is the number of test instances, and in common case $q \ll n$. Therefore, the test computation complexity of RDT (include LP-RDT and BR-RDT) is $O(mnq)$. That means both the test computation complexities of LP-RDT and BR-RDT are low and unrelated to the number of class labels $|L|$. Compared with BR-RDT, other BR multi-label classifiers need test each label respectively. That results the test computation complexity of them depends on $|L|$. HOMER algorithm reduces the test computation complexity of BR methods from $O(|L|)$ to $O(\log_k(|L|))$, where k is the number of subsets of labels. The test complexity of HOMER algorithm still depends on $|L|$.

5 Experiments

In previous sections, we theoretically proved that LP-RDT and BR-RDT are robust and efficient algorithms for multi-label classification. The reliability is based on the stability of the learning risk, and the efficiency comes from their training complexities are independent from the number of class labels. In this section, we will further explore the two aspects of LP-RDT and BR-RDT by doing experiments on several typical real datasets, and compare them with most popular algorithms in this domain. At last, we examine the capability of BR-RDT on the problem of large number of labels and compare it with HOMER algorithm [12].

5.1 Datasets We select five small datasets (genbase, yeast, scene, emotions, and enron) and one large dataset (delicious) from Mulan site [11], and make sure they cover different domains and carry different statistic characteristics. Among them, genbase and yeast are biological datasets with regard to protein function classification and gene function classification respectively. In genbase, each instance means a protein and each label means a protein class (Prosite documentation ID), and every protein can belong to more than one classes. For the yeast data, each instance is a gene associated with a set of functional classes whose maximum size can be more than 190. The dataset scene comes from the image categorization problem, and each instance represents an image with multiple class labels, e.g., an image can be labeled as Beach and Mountain at the same time. Emotion is from music emotion classification, and each instance represents a song and a label is the genre of the song, like classical, pop, rock etc. Enron is a subset of Entro email dataset, every instance is an email and the label is its categorization. Delicious is extracted from delicious.us, and each instance is a web page and a label is a tag. Delicious is much larger than other five datasets. With limit memory and computation resource, the selected algorithms except HOMER [12] can't run on delicious, so we only use it to compare ML-RDT with HOMER. Table 2 shows some detailed information about selected datasets, such as the number of instances, number of numeric and discrete attributes, number of class labels and total number of distinct combinations of labels appeared in the dataset. The cardinality is the average number of labels for each instance, and the density is defined as the division of cardinality by the number of labels.

5.2 Evaluation Metrics Multi-label classification uses different metrics than traditional single-label classification. Here we introduce several popular metrics that have been widely used in the literature [8] to evaluate multi-label classifiers.

Let D be a multi-label dataset, consisting of $|D|$ multi-label examples (x_i, Y_i) , where $i = 1, \dots, |D|$ and

Table 3: Table of comparison between RDT and other classifiers

Metric	Scene dataset							
	BR-KNN	BR-C4.5	BR-NB	BR-SMO	ML-KNN	LP-RDT	BR-RDT	RAKEL
HamLoss	0.125	0.139	0.247	0.114	0.087	0.080	0.084	0.092
Accuracy	0.637	0.513	0.435	0.571	0.675	0.751	0.737	0.636
Recall	0.669	0.534	0.443	0.596	0.692	0.751	0.737	0.662
Precision	0.651	0.611	0.816	0.628	0.702	0.788	0.775	0.658
Time	428.549s	15.453s	4897.799s	28.119s	36.983s	4.064s	21.429	1995.262s
Metric	Emotion dataset							
	BR-KNN	BR-C4.5	BR- NB	BR-SMO	ML-KNN	LP-RDT	BR-RDT	RAKEL
HamLoss	0.260	0.260	0.256	0.215	0.294	0.200	0.225	0.226
Accuracy	0.487	0.438	0.420	0.483	0.319	0.603	0.474	0.492
Recall	0.600	0.568	0.509	0.543	0.377	0.739	0.504	0.583
Precision	0.595	0.586	0.563	0.629	0.502	0.667	0.709	0.620
Time	6.902s	1.202s	104.954s	1.125s	0.797s	1.142s	6.124s	52.000
Metric	Genbase dataset							
	BR-KNN	BR-C4.5	BR-NB	BR-SMO	ML-KNN	LP-RDT	BR-RDT	RAKEL
HamLoss	0.000	0.001	0.035	0.000	0.005	0.001	0.002	-
Accuracy	0.989	0.987	0.273	0.991	0.944	0.987	0.973	-
Recall	0.997	0.995	0.276	0.997	0.946	0.988	0.973	-
Precision	0.992	0.992	0.273	0.993	0.974	0.997	0.999	-
Time	1327.394s	3.890s	308.319s	83.560	12.524s	28.445s	36.983s	-
Metric	Yeast dataset							
	BR-KNN	BR-C4.5	BR-NB	BR-SMO	ML-KNN	LP-RDT	BR-RDT	RAKEL
HamLoss	0.243	0.259	0.301	0.200	0.194	0.212	0.210	-
Accuracy	0.479	0.423	0.421	0.502	0.510	0.529	0.417	-
Recall	0.601	0.593	0.531	0.711	0.578	0.624	0.437	-
Precision	0.596	0.561	0.610	0.579	0.726	0.659	0.748	-
Time	535.797s	27.040s	696.627s	47.315s	16.444s	13.996s	74.302s	-
Metric	Enron dataset							
	BR-KNN	BR-C4.5	BR-NB	BR-SMO	ML-KNN	LP-RDT	BR-RDT	RAKEL
HamLoss	0.068	0.054	-	0.057	0.051	0.023	0.048	-
Accuracy	0.304	0.366	-	0.406	0.319	0.386	0.403	-
Recall	0.389	0.446	-	0.520	0.358	0.463	0.438	-
Precision	0.435	0.570	-	0.566	0.587	0.533	0.711	-
Time	1143.246s	27.416s	-	12.794s	11.350s	22.646s	50.582s	-

$Y_i \subset L$. Let h be a multi-label classifier and $Z_i = h(x_i)$ be the set of labels predicted by h for the example x_i .

Hamming Loss [23], which considers the prediction error (an incorrect label is predicted) and missing error (a label is not predicted) at the same time, is defined as:

$$\text{HammingLoss}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \Delta Z_i|}{|L|}$$

where Δ means the symmetric difference of two sets and corresponds to the XOR operation in bool logic. The smaller the value of Hamming Loss, the better the performance.

The following metrics are used in [24], for evaluation of h on D :

$$\text{Accuracy}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

$$\text{Recall}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$\text{Precision}(h, D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Y_i \cap Z_i|}{|Z_i|}$$

The higher the value of accuracy, precision, and recall, the better the performance. As accuracy also takes into account prediction error and missing error, we consider accuracy and hamming loss as two major metrics.

5.3 Experimental Results As follows, we evaluate ML-RDT from two aspects: effectiveness and efficiency.

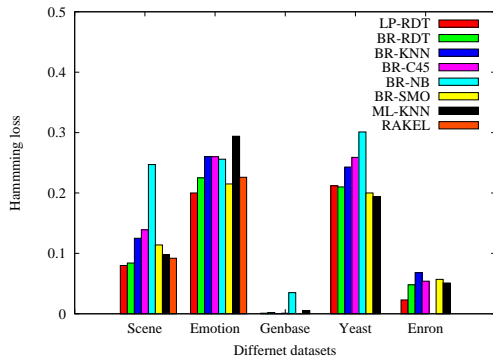


Figure 2: Errors change with the increasing of the number of trees on the scene dataset

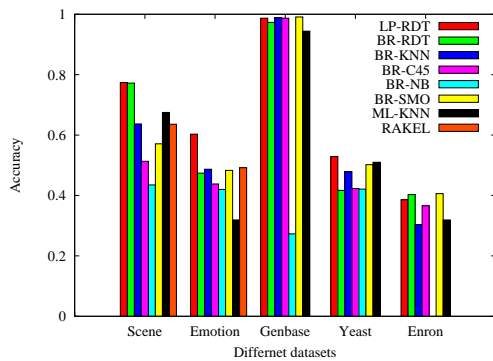


Figure 3: Errors change with the increasing of the number of trees on the scene dataset

At first, we compare LP-RDT and BR-RDT to other main stream algorithms on five small datasets mainly for effectiveness comparison. After that, to evaluate efficiency, we compare BR-RDT with a number of algorithms on the artificial datasets with large number of labels or instances. At last, we compare BR-RDT with HOMER algorithm on delicious dataset, because only the HOMER [12] from selected algorithms can handle delicious.

5.3.1 Results on Small Datasets In this subsection, we compare LP-RDT and BR-RDT to BR-KNN (K-Nearest Neighbor), BR-C4.5 [1], BR-NB (Naive Bayesian), BR-SMO, ML-KNN [20] and RAKEL [14]. SMO [15] is a method to train SVM fast. We use 5-fold cross validation on the experiments to evaluate algorithms' effectiveness and efficiency. In order to balance performance and efficiency, we construct 200 trees for BR-RDT and LP-RDT on all the experiments, the maximum depth is the half of the number of attributes, and the minimal instances on a leaf node is 4.

In Table 3, there are 4 effectiveness metrics and the

time spent during the cross-validation. The 4 metrics results of BR-KNN, BR-C4.5, BR-NB, BR-SMO on scene, genbase, and yeast data are extracted from [13], others are ran by using Weka [6] and Mulan [11] implementations, and we also implemented BR-RDT and LP-RDT algorithms based on these frameworks. Because RAKEL algorithm can not accomplish the calculation in 8 hours on genbase, yeast and enron, no results were provided here. It is the same for NB on enron dataset.

We use Hamming Loss and Accuracy as our primary indicators. On the scene dataset, LP-RDT ranks first on all of metrics, BR-RDT is as good as LP-RDT in hamming loss, and a little weaker in accuracy; ML-KNN comes to the third place, with 13% weaker in hamming loss and 12.8% in accuracy than the LP-RDT. On emotion data, LP-RDT outperforms 18.4% than RAKEL which ranks second in accuracy, and outperforms 7.5% than BR-SMO which ranks second in hamming loss. On enron dataset, LP-RDT and BR-RDT perform best in hamming loss and precision respectively, although SMO ranks first in accuracy (0.406), BR-RDT follows closely behind (0.403), while the precision of BR-RDT is 25.6% higher than BR-SMO. In a word, on these three datasets, LP-RDT and BR-RDT have quite obviously advantages over other algorithms, especially LP-RDT, shows compelling results on scene and emotion data. On genbase data, BR-SMO performs best in both hamming loss and accuracy, while LP-RDT and BR-RDT fall behind slightly with less than 1%; while in precision, these two are slightly better than BR-SMO. On yeast, ML-KNN performs best in hamming loss, while LP-RDT ranks first in accuracy. So we can say on these two biology datasets, LP-RDT and BR-RDT still have comparatively good results.

Figure 2 and Figure 3 illustrates an overview of effectiveness comparison between LP-RDT and BR-RDT and other methods in multi-label classification. It can be seen that LP-RDT and BR-RDT, especially LP-RDT outperforms others obviously on scene, emotion and enron; for the other data sets, ML-RDT algorithms continue their steady performance.

Figure 4 gives a comparison putting hamming loss and time cost together, showing each algorithm's computation time and hamming loss on the original five datasets from Table 2. In this scatter plot, the vertical axis represents the computation time and horizon axis represents the hamming loss. Each algorithm is represented by an icon with different shape. According to the meaning of the axis, the points in the bottom left means both fast in speed and good in prediction accuracy, while the upper right corner means the algorithm

Table 4: Performance of BR-RDT on delicious dataset

Tree Number	Training Time	Test Time	Hamming Loss
10	14.687s	18.531s	0.02039
30	43.047s	77.485s	0.02037

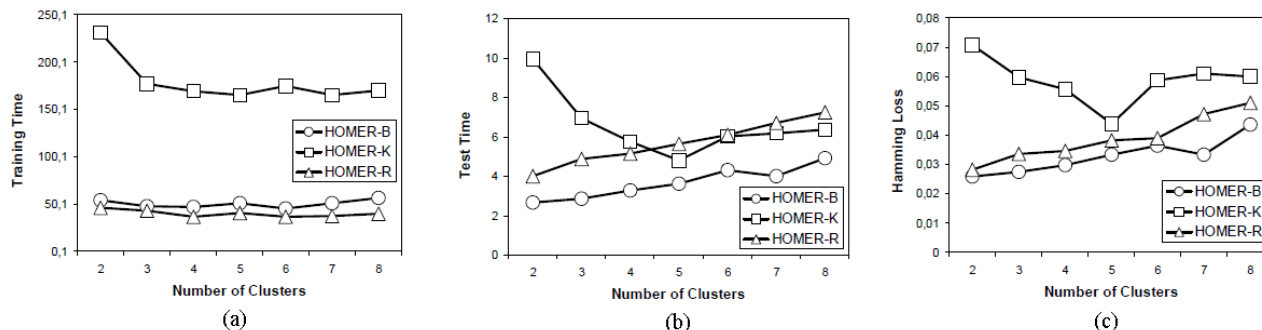


Figure 5: Performance of HOMER on delicious dataset

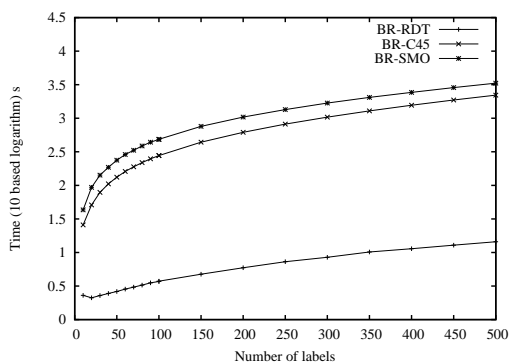


Figure 6: Computation time changes as the number of labels increase

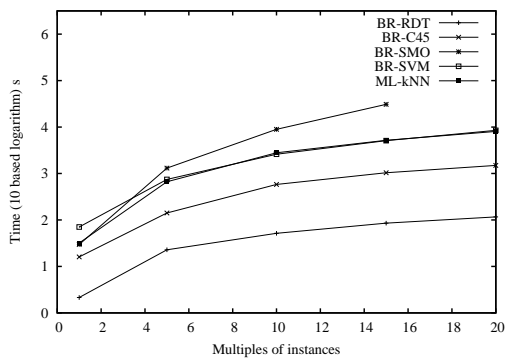


Figure 7: Computation time changes as the data size increase

Second, we examine the relationship between computation time and the number of instances. We generate the dataset with different size by copying the whole instances of scene multiple times. In this experiment, the classic classifiers like BR-KNN, BR-C4.5, and BR-SMO are used for comparison. The results are illustrated by Figure 7, the vertical axis of which is 10 based logarithm of computation time in second. From original size to 20 times large, BR-RDT is the fastest method all the time. More importantly, the growth rate of computation time of BR-RDT is also the slowest one among all of them. Comparatively, the computation time of other methods grows rather fast, especially BR-SMO, which is very sensitive to the growth of the size of instances. For 48,000 instances, BR-RDT takes 117s, BR-C4.5 takes 1492s, and ML-KNN takes 8060s. For 36,000 instance, BR-SMO costs 30836s, and can not finish in a long time when the number of instances reaches 48,000.

Third, we compare the BR-RDT with HOMER on delicious. The delicious dataset was used by Tsoumakas et al. [12] to evaluate the HOMER algorithm. The HOMER algorithm is developed to solve the problem of large number of labels. From Table 2, delicious dataset contains 983 labels, and 12920 and 3185 instances in training and test set respectively, the numbers of instances are larger than all other datasets.

Here, we evaluate the effectiveness and efficiency and compare RDT methods with HOMER. Because the number of distinct label combinations is 15806, LP-RDT method is not appropriate for delicious. We only use BR-RDT method. We train a batch of trees on training dataset of delicious and test those trees on test dataset of delicious. The minimal number of instances

on a leaf node of the tree is 4 and the maximum depth is 500, the same with the number of attributes.

Table 4 shows the results of two BR-RDT experiments with 10 and 30 trees respectively. Using 10 trees, training time is 14.687 seconds, test time is 18.531 seconds and the hamming loss is 0.02039. Using 30 trees, training time is 43.047 seconds, test time is 77.485 seconds and the hamming loss is 0.02037. In this experiment, we use a desktop with Intel Core 2 6700 @2.66GHz CPU, 2G RAM and Windows XP operating system. Next, we introduce the results of HOMER on delicious. In Figure 5, all the 3 sub-figures are directly copied from [12], in which the time unit is minute. Figure 5 (a) shows the training time of HOMER and variations are all greater than 40 minutes. Figure 5 (b) shows the test time of HOMER and variations are greater than 2 minutes. Moreover, the hamming loss of HOMER and variations are greater than 0.024 in Figure 5 (b). Comparing the performance of BR-RDT with HOMER, the efficiency of BR-RDT is much better than HOMER. Especially for training process, BR-RDT runs at least 2 to 3 orders of magnitude faster than HOMER and variations. The effectiveness of BR-RDT is also better than HOMER. Note, due to the lack of descriptions of the environment of HOMER's results, the comparison of BR-RDT with HOMER is not precise but somewhat convinced.

Apparently, BR-RDT has significant advantages on the problems with large number of labels and instances. That is why we claim it is the method without multi-label cost.

6 Related Work

In recent years, many approaches have been developed to solve the multi-label classification problem. Tsoumakas and Katakis [9] summarized them into two categories: problem transformation and algorithm adaptation.

A simple problem transformation method transforms all the appeared combinations of the labels in the training data to new labels. As the result, the size of the transformed labels is at most $2^{|L|}$. It is also called label powerset (LP) [21]. To solve the problem of the large size of label combinations, [17] developed pruned problem transformation method (PPT), which only selects the transformed labels that occur more than pre-defined threshold. The random k-labelsets (RAKEL) [14] was proposed to take the correlation of labels into consideration to improve the classification. Binary Relevance (BR) [21] is another popular problem transformation method. SVM is a popular classifier adopted by BR methods [21, 28]. The SMO [15] is a fast algorithm to train SVM used in [13].

There are abundant algorithm adaptation methods. [1] adapted the C4.5 algorithm by modifying the entropy calculation. Probabilistic generative models for multi-label classification were studied in [2, 22]. Neural networks were also adapted for multi-label classification in [20, 18]. Several methods [19, 3, 5] were derived from k Nearest Neighbours lazy learning algorithm. Association rule mining method was also adapted in [7]. Two boosting methods Adaboost.MH and Adaboost.MR were proposed in [23].

Besides those above methods for classical multi-label classification problem, Celine Vens and Jan Struyf [4] developed some methods to solve the Hierarchical multi-label classification problem(HMC). Which is a variant of classification where instances may belong to multiple classes at the same time and these classes are organized in a hierarchy.

7 Conclusions

In this paper, we studied how to adopt and improve Random Decision Tree to solve the efficiency and robustness problems for multi-label classification. We formally analyzed the learning risk of Random Decision Tree (RDT), which guarantees good and robust performance on different applications. We also found that the computation complexity of ML-RDT is much lower than other famous decision tree algorithms. Importantly, the computation complexity of ML-RDT is independent of the number of labels, so that ML-RDT can effectively handle large number of labels. The experimental results demonstrated that ML-RDT outperform other popular algorithms in accuracy, reliability and efficiency: ML-RDT improves up to 10% in accuracy compared to a few state-of-the-art multi-label classifiers; for reliability, it is always in the top rank across different datasets; and in efficiency, it runs 1-3 orders of magnitude faster than other methods while the number of labels is 500 or the size of dataset is larger than 48,000. In addition, BR-RDT is much faster and more accurate than HOMER. Especially on the training process, it runs 2-3 orders of magnitude faster than HOMER.

References

- [1] A. Clare, A. Clare and R. D. King, *Knowledge Discovery in Multi-Label Phenotype Data*, Proceedings of PKDD 2001, 2001.
- [2] A. K. McCallum, *Multi-label text classification with a mixture model trained by EM*, AAAI 99 Workshop on Text Learning, 1999.
- [3] A. Wiczorkowska, P. Synak and Z. Raś, *Multi-Label Classification of Emotions in Music*, in Intelligent Information Processing and Web Mining, 2006, pp. 307–315.

- [4] C. Vens and J. Struyf, *Decision Trees for Hierarchical Multi-label Classification*, Machine Learning, Volume 73 ,Issue 2 (November 2008), pp. 185–214.
- [5] E. Spyromitros, G. Tsoumakas and I. Vlahavas, *An Empirical Study of Lazy Multilabel Classification Algorithms*, in Artificial Intelligence: Theories, Models and Applications, 2008, pp. 401–406.
- [6] E. Frank and M. Hall and T. Trigg, *Weka*, <http://www.cs.waikato.ac.nz/ml/weka/>.
- [7] F. A. Thabtah, P. Cowling and Y. Peng, *MMAC: A New Multi-class, Multi-label Associative Classification Approach*, IEEE ICDM, 4 (2004).
- [8] G. Tsoumakas and I. Katakis, *Multi-Label Classification: An Overview*, Dept. of Informatics, Aristotle University of Thessaloniki, Greece.
- [9] G. Tsoumakas and I. Katakis, *Multi-Label Classification: An Overview*, in International Journal of Data Warehousing and Mining, 2007.
- [10] G. Tsoumakas, I. Katakis and I. Vlahavas, *draft of preliminary accepted chapter*, in Data Mining and Knowledge Discovery Handbook (2nd), Springer, 2009.
- [11] G. Tsoumakas and R. Friberg, E. Spyromitros-Xioufis and I. Katakis and J. Vilcek, *Multi-Label Classification*, <http://mlkd.csd.auth.gr/multilabel.html>.
- [12] G. Tsoumakas, I. Katakis and I. Vlahavas, *Effective and Efficient Multilabel Classification in Domains with Large Number of Labels*, ECML/PKDD 2008 Workshop on Mining Multidimensional Data, 2008.
- [13] G. Tsoumakas and I. Katakis, *Multi Label Classification: An Overview*, International Journal of Data Warehouse and Mining, Idea Group Publishing, 3 (2007), pp. 1–13.
- [14] G. Tsoumakas and I. Vlahavas, *Random k-Labelsets: An Ensemble Method for Multilabel Classification*, ECML PKDD, 18 (2007).
- [15] J. C. Platt, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, 1998.
- [16] J. K. Martin and D. S. Hirschberg, *The Time Complexity of Decision Tree Induction*, 1995.
- [17] J. Read, *A pruned problem transformation method for multi-label classification*, NZCSRS, 2008.
- [18] K. Crammer, Y. Singer, J. K. T. Hofmann, T. Poggio and J. Shawe-taylor, *A Family of Additive Online Algorithms for Category Ranking*, in Journal of Machine Learning Research, 3 (2003).
- [19] M. L. Zhang and Z. H. Zhou, *Ml-knn: A lazy learning approach to multi-label learning*, in Pattern Recognition, 40 (2007).
- [20] M. L. Zhang and Z. H. Zhou, *Multi-Label Neural Networks with Applications to Functional Genomics and Text Categorization*, IEEE Transactions on Knowledge and Data Engineering, 18 (2006), pp. 1338–1351.
- [21] M. R. Boutell, J. Luo, X. Shen and C. M. Brown, *Learning multi-label scene classification*, in Pattern Recognition, 37 (2004), pp. 1757–1771.
- [22] N. Ueda and K. Saito, *Parametric mixture models for multi-labeled text*, NIPS, 15 (2003), pp. 721–728.
- [23] R. E. Schapire and Y. Singer, *BoosTexter: A Boosting-based System for Text Categorization*, in Machine Learning, 39 (2000), pp. 135–168.
- [24] S. Godbole and S. Sarawagi, *Discriminative Methods for Multi-labeled Classification*, Advances in Knowledge Discovery and Data Mining, AAAI, 2004, pp. 22–30.
- [25] W. Fan, *StreamMiner: A Classifier Ensemble-based Engine to Mine Concept-drifting Data Streams*, VLDB, 30 (2004), pp. 1338–1351.
- [26] W. Fan, J. McCloskey and P. S. Yu, *A General Framework for Accurate and Fast Regression by Data Summarization in Random Decision Trees*, ACM SIGKDD, 12 (2006), pp. 136–146.
- [27] W. Fan, H. Wang, P. S. Yu and S. Ma, *Is random model better? On its accuracy and efficiency*, IEEE ICDM, 3 (2003).
- [28] Z. H. Zhou and M. L. Zhang, *Multi-Instance Multi-Label Learning with Application to Scene Classification*, NIPS, 20 (2006), pp. 1609–1616.