ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200274

Multi-Label Learning with Deep Forest

Liang Yang and **Xi-Zhu Wu** and **Yuan Jiang** and **Zhi-Hua Zhou** [1]

Abstract. In multi-label learning, each instance is associated with multiple labels, and the crucial task is how to leverage label correlations in building models. The deep forest is a recent deep learning framework based on decision tree ensembles, which has a cascade structure that can do representation learning like deep neural models and does not rely on backpropagation. Though deep forests have been found useful in classification tasks, the potential of applying it into multi-label learning has not been studied. We consider that the layer-by-layer processing structure of the deep forest is appropriate for solving multi-label problems. Therefore we design the Multi-Label Deep Forest (MLDF) method, including two mechanisms: measure-aware feature reuse and measure-aware layer growth. The measure-aware feature reuse mechanism enables MLDF to reuse better representation in the previous layer. The measure-aware layer growth mechanism ensures MLDF gradually increase the model complexity guided by performance measure. MLDF handles two challenging problems at the same time: one is restricting the model complexity to ease the overfitting issue; another is optimizing the performance measure on user's demand since there are many different measures in the multi-label evaluation. Experiments demonstrate that our proposal not only beats the compared methods over six measures on benchmark datasets but also enjoys label correlation discovery and other desired properties in multi-label learning.

1 INTRODUCTION

In multi-label learning, each instance is associated with multiple labels simultaneously, and the task is to predict a set of relevant labels for unseen instances. Multi-label learning has been widely applied in diverse problems like text categorization [31], scene classification [30], functional genomics [26], video categorization [17], chemicals classification [4], etc, and multi-label learning tasks are omnipresent in real-world problems [22].

By transforming the multi-label learning problem to independent binary classification problems for each label, Binary Relevance [21] is a widely-used straightforward method. Though it is intended to make full use of high-performance traditional single-label classifiers, it will lead to high computational cost when label space is enormous. Besides, such a method neglects the fact that information on one label may help learn other related labels, which would limit the prediction performance. Investigating correlations among labels has been demonstrated to be crucial to improve the performance of multi-label learning. As a result, more and more multi-label learning methods aimed to explore and exploit the label correlations are proposed [22]. There emerges considerable attention to explore and exploit label correlations in multi-label learning methods [22, 24, 28].

Different from traditional multi-label methods, deep neural network models usually make an effort to learn a new feature space and employ a multi-label classifier on the top. Among the first to utilize network architectures, BP-MLL [26] not only treats each output node as a binary classification task but also exploits label correlations relied on the architecture itself. Later, a comparably simple neural network approach builds upon BP-MLL was proposed by replacing the pairwise ranking loss with entropy loss [13]. It achieves a good result in the large-scale text classification. However, deep neural models usually require a massive amount of training data, and thus they are not usually suitable for small-scale datasets.

By realizing that the essence of deep learning lies in layer-by-layer processing, in-model feature transformation, and sufficient model complexity, Zhou and Feng proposed deep forest [34]. The deep forest is a deep ensemble model built on decision trees and does not use backpropagation in the training process. A deep forest ensemble with a cascade structure can do representation learning like deep neural models. Compared to DNN, the deep forest is much easier to train since it has fewer hyperparameters. It has achieved excellent performance on a broad range of tasks, such as large-scale financial fraud detection [29], and it has been found that forest models can achieve some important properties that were believed to be owned only by neural networks, such as the auto-encoder ability [7] and hierarchical distributed representation ability [6]. Though deep forest has been found useful in classification tasks [34], the potential of applying it into multi-label learning has not been studied before our work.

The success of deep forest mainly comes from the layer-by-layer feature transformation in an ensemble way [32]. While on the other hand, the critical point in multi-label learning is how to take advantage of label correlations. Inspired by these two facts, we propose the Multi-Label Deep Forest (MLDF) method. Briefly speaking, MLDF uses different multi-label tree methods as the building blocks in deep forest, and label correlations can be exploited via layer-by-layer representation learning.

Because evaluation in multi-label learning is more complicated than traditional classification tasks, various performance measures have been proposed [20]. It is worthy to notice that different users have different demands, and an algorithm usually performs differently on different measures [25]. To achieve better performance on the specific measure, we propose two mechanisms: measure-aware feature reuse and measure-aware layer growth. Inspired by confidence screening [14], we define the confidence of each multi-label measure considered in our work. The measure-aware feature reuse mechanism reuses better presentation in the previous layer for specific measures when the confidence is higher than the threshold. The measure-aware layer growth mechanism aims to control the model complexity by various performance measures. The two measure-aware mechanisms promote the performance of MLDF.

The main contributions of this paper are summarized as follows:

[1] National Key Laboratory for Novel Software Technology, Nanjing University, China, email: {yangl, wuxz, jiangy, zhouzh}@lamda.nju.edu.cn

- We first introduce the deep forest to multi-label learning, which provides another effective approach to deep multi-label learning.
- We propose the measure-aware feature reuse mechanism based on confidence computing. It can optimize different performance measures on user's demand.
- We take the measure-aware layer growth mechanism to enable MLDF to reduce overfitting when utilizing label correlations by a large number of layers, which often observed in deep neural multi-label models.
- Our extensive experiments show that MLDF achieves the best performance on nine benchmark datasets over six multi-label performance measures. Furthermore, investigative experiments demonstrate that our proposal enjoys high flexibility in applying various base tree models.

The remainder of the paper is organized as follows. Section 2 introduces some preliminaries. Section 3 formally describes our MLDF method, including two measure-aware mechanisms. Section 4 reports the experimental results on benchmarks and investigative experiments. Finally, we conclude the paper in Section 5.

2 PRELIMINARIES

In this section, we introduce various multi-label performance measures, followed by tree-based multi-label methods, which will be used in our proposal.

2.1 Multi-label performance measures

The multi-label classification task is to derive a function H from the training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)|1 \leq i \leq m, \boldsymbol{x}_i \in \mathbb{R}^d, \boldsymbol{y}_i \in \{0,1\}^l\}$. Suppose the multi-label learning model first produces a real-valued function $f : \mathcal{X} \to [0,1]^l$, which can be viewed as the confidence of relevance of the labels. The multi-label classifier $h : \mathcal{X} \to \{0,1\}^l$ can be induced from f by thresholding.

There are lots of performance measures for multi-label learning. Six widely-used evaluation measures [25] are employed in this paper. Table 1 shows the formulation of these measures, being Y the true labels, $Y_{i\cdot}$ the i-th row of the label matrix, '+'('−') the relevant (irrelevant) note. Hamming loss and macro-AUC are label-based measures, while one-error, coverage, ranking loss, and average precision are instance-based measures [27]. The f_{ij} means the confidence score of i-th instance on j-th label, h_{ij} means the predicted result of i-th instance on j-th label, and $\mathrm{rank}_f(\boldsymbol{x}_i, j)$ means the instance $\boldsymbol{x}_i$'s rank on j-th label. For example, if $\boldsymbol{f}(\boldsymbol{x_i}) = [0.2, 0.8, 0.4]$, then we have $\boldsymbol{h}(\boldsymbol{x_i}) = [0,1,0]$ with threshold 0.5. Furthermore, we have $Y_{i\cdot}^- = \{1,3\}$ and $Y_{i\cdot}^+ = \{2\}$. Since $f_{i2} = 0.8$, then we have $\mathrm{rank}_f(\boldsymbol{x}_i, 2) = 1$.

2.2 Tree-based multi-label methods

Tree-based multi-label methods, such as ML-C4.5 [5] and PCT [2], are adapted from multi-class decision tree methods. They allow multiple labels in the leaves of the tree, whose formula is modified by summing the criterion value of each label. The information kept in each leaf node is the probability that the instance owns each label by counting the percentage of different classes of training examples at the leaf node where concerned instance falls. In the testing process of both methods, the leaf node of a multi-label tree returns a vector of probabilities that a sample belongs to each class.

The learning ability of a single tree is limited, and the ensemble of the trees will significantly improve performance. Random Forest of Predictive Clustering Trees (RF-PCT) [11] and Random Forest of ML-C4.5 (RFML-C4.5) [12] are ensembles that use PCT and ML-C4.5 as base classifiers respectively. The same as random forest, these forests use bagging and choose different feature sets to obtain the diversity among the base classifiers. The number of features in retained subsets can be configured on the square root of the number of original features. Given a test instance, the forest will produce an estimate of label distribution by averaging results across all trees.

Table 1. Definitions of six multi-label performance measures. '↓' means the lower the better, '↑' means the higher the better.

Measure	Formulation
hamming loss ↓	$\frac{1}{ml}\sum_{i=1}^{m}\sum_{j=1}^{l}\mathbb{I}[h_{ij} \neq y_{ij}]$
one-error ↓	$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left[\arg\max f(\boldsymbol{x}_i) \notin Y_{i\cdot}^+\right]$
coverage ↓	$\frac{1}{ml}\sum_{i=1}^{m}\mathbb{I}\left[\max_{j\in Y_{i\cdot}^+}\mathrm{rank}_f(\boldsymbol{x}_i, j) - 1\right]$
ranking loss ↓	$\frac{1}{m}\sum_{i=1}^{m}\frac{\lvert\mathcal{S}_{\mathrm{rank}}^i\rvert}{\lvert Y_{i\cdot}^+\rvert\lvert Y_{i\cdot}^-\rvert}$
average precision ↑	$\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\lvert Y_{i\cdot}^+\rvert}\sum_{j\in Y_{i\cdot}^+}\frac{\lvert\mathcal{S}_{\mathrm{precision}}^{ij}\rvert}{\mathrm{rank}_f(\boldsymbol{x}_i,j)}$
macro-AUC ↑	$\frac{1}{l}\sum_{j=1}^{l}\frac{\lvert\mathcal{S}_{\mathrm{macro}}^j\rvert}{\lvert Y_{\cdot j}^+\rvert\lvert Y_{\cdot j}^-\rvert}$

$\mathcal{S}_{\mathrm{rank}}^i = \{(u,v) \in Y_{i\cdot}^+ \times Y_{i\cdot}^- | f_u(\boldsymbol{x}_i) \leq f_v(\boldsymbol{x}_i)\}$

$\mathcal{S}_{\mathrm{precision}}^{ij} = \left\{k \in Y_i^+ | \mathrm{rank}_f(\boldsymbol{x}_i, k) \leq \mathrm{rank}_f(\boldsymbol{x}_i, j)\right\}$

$\mathcal{S}_{\mathrm{macro}}^j = \left\{(a,b) \in Y_{\cdot j}^+ \times Y_{\cdot j}^- | f_j(\boldsymbol{x}_a) \geq f_j(\boldsymbol{x}_b)\right\}$

3 THE PROPOSED METHOD

In this section, we propose a deep forest method for multi-label learning. Firstly, we introduce the general framework of Multi-Label Deep Forest (MLDF). Then, we explain two proposed mechanisms in detail: measure-aware feature reuse and measure-aware layer growth.

3.1 The framework

Figure 1 illustrates the framework of MLDF. Different multi-label forests (the black forests above and the blue forests below) are ensembled in each layer of MLDF. From layer_t, we can obtain the representation $\mathbf{H}^t$. The part of measure-aware feature reuse will receive the representation $\mathbf{H}^t$ and update it by reusing the representation $\mathbf{G}^{t-1}$ learned in the layer_{t-1} under the guidance of the performance of different measures. Then the new representation $\mathbf{G}^t$ (the green one) will be concatenated together with the raw input features (the red one) and goes into the next layer.

In MLDF, each layer is an ensemble of forests. To enhance the performance of the ensemble, we consider different growing methods of trees to encourage diversity, which is crucial to the success of ensemble methods [32]. In traditional multi-class problems, extremely-random trees [8], which takes one split point of each feature randomly, are used in gcForest [33]. For multi-label learning problems, we can also adopt this kind of method by changing the approach of splitting nodes when generating trees. In MLDF, we take RF-PCT

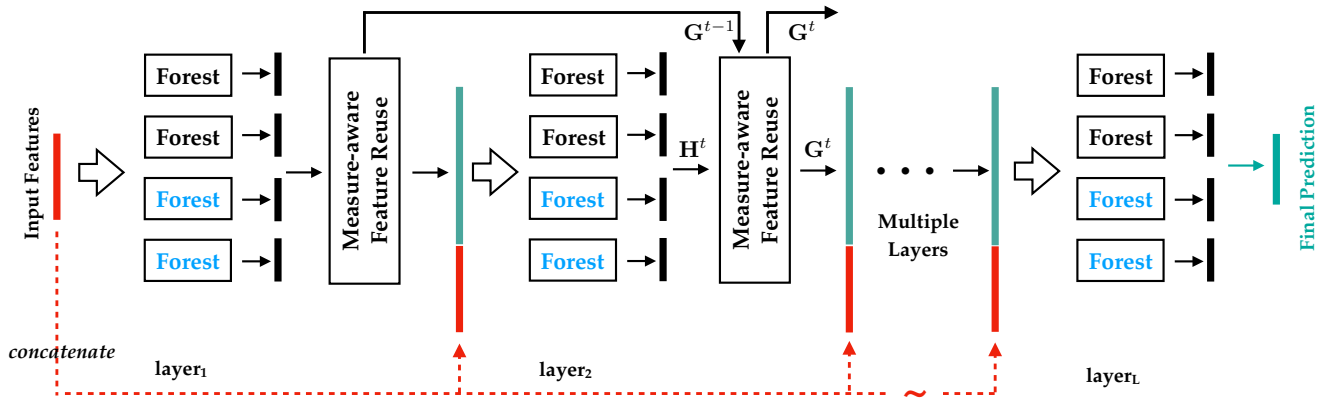

Figure 1. The framework of Multi-Label Deep Forest (MLDF). Each layer ensembles two different forests (the black above and the blue below).

Algorithm 1 Measure-aware feature reuse

Input: measure M, forests' output $\mathbf{H}^t$, previous $\mathbf{G}^{t-1}$.
Output: new representation $\mathbf{G}^t$.
Procedure:

```
Initialize tensor G^t = H^t.
if Measure M is label-based then
    for j = 1 to l do
        compute confidence α_j^t on H_{:j}^t
        Update G_{:j}^t to G_{:j}^{t-1} when α_j^t < θ_t.
    end for
end if
if Measure M is instance-based then
    for i = 1 to m do
        compute confidence α_i^t on H_{i:}^t
        Update G_{i:}^t to G_{i:}^{t-1} when α_i^t < θ_t.
    end for
end if
```

[11] as the forest block, and two different methods generating nodes in trees are used to forests: one considers all possible split points of each feature, which is RF-PCT (the black one), the other considers one split point randomly [10], we name this as ERF-PCT (the blue one). Of course, other multi-label tree methods can also be embedded in each layer, such as RFML-C4.5.

Assume all the basic forests have been fitted, as shown in Figure 1, the predicting process can be summarized as follows. Firstly, we pre-process instances to standard matrix $\mathbf{X}$. Secondly, the instances matrix $\mathbf{X}$ passes the first layer. As aforementioned in Section 2.2, given an instance, the forests will produce an estimation of label distribution, which can be viewed as the confidence of the instance belonging to each label. Therefore, we can get the representation $\mathbf{H}^1$. By adopting measure-aware feature reuse, we can get $\mathbf{G}^1$. Then we concatenate $\mathbf{G}^1$ with the raw input features $\mathbf{X}$ and put them into the next layer. The real-valued representation $\mathbf{G}^1$ of rich labeling information will be considered in the next layer to facilitate MLDF to take better advantage of label correlations [1]. After multiple layers, we obtain the final prediction.

3.2 Measure-aware feature reuse

The split criterion of PCT is not directly related to the performance measure, and the representation tensor $\mathbf{H}^t$, whose shape is $(\#\text{num\_forests}, \#\text{num\_samples}, \#\text{num\_labels})$, in layer$_t$ during training is agnostic to the measure we want to optimize. Therefore we propose the measure-aware feature reuse mechanism to enhance the representation under the guidance of different measures. The critical idea of measure-aware feature reuse is to partially reuse the better representation in the previous layer on the current layer if the confidence on the current layer is lower than a threshold determined in training. Therefore, the challenge lies in defining the confidence of specific multi-label measures on demand. Inspired by the confidence corresponding to accuracy in the multi-class classification problem [14], we define the confidence in all six multi-label measure.

Table 2. Confidence computing method on six multi-label measures. $\boldsymbol{p}_{i\cdot}$ or $\boldsymbol{p}_{\cdot j}$ is sorted in descending order. * represents the instance-based measure and $^\diamond$ represents the label-based measure.

Measure	Confidence
$^\diamond$hamming loss	$\frac{1}{m}\sum_{i=1}^{m} p_{ij}\mathbb{I}[p_{ij} > 0.5] + (1-p_{ij})\mathbb{I}[p_{ij} \le 0.5]$
*one-error	$\max_{j=1,\ldots,l} p_{ij}$
*coverage	$1 - \frac{1}{l}\sum_{j=0}^{l}\left[j \cdot p_{ij} \prod_{k=j+1}^{l}(1-p_{ik})\right]$
*ranking loss	$\sum_{j=0}^{l}\prod_{k=1}^{j} p_{ik} \prod_{k=j+1}^{l}(1-p_{ik})$
*average precision	$\sum_{j=0}^{l}\prod_{k=1}^{j} p_{ik} \prod_{k=j+1}^{l}(1-p_{ik})$
$^\diamond$macro-AUC	$\sum_{i=0}^{m}\prod_{k=1}^{i} p_{kj} \prod_{k=i+1}^{m}(1-p_{kj})$

Comparing the prediction matrix with the true label matrix, hamming loss cares the correctness of single bit, one-error cares the element closest to 1, and others care the ranking permutation on each row or column. In general, label-based measure and instance-based measure are quite different[27], therefore we handle them separately. Table 2 summarizes the computing method by considering the inherent meaning of each measure. Matrix $\mathbf{P}$ is the average of $\mathbf{H}^t$ on the first dimension, and the element p_{ij} represents $\Pr[\hat{y}_{ij} = 1]$. Without loss of generality, we rearrange the elements of each row (column) of $\mathbf{P}$ in descending order when the measure is instance-based (label-based). Explicitly, for hamming loss, we compute the max confidence that the bit is positive or negative. For example, the prediction vector $\mathbf{p}_{\cdot j} = [0.9, 0.6, 0.4, 0.3]$, thus the confidence is

Algorithm 2 Determine threshold

Input: measure M, forests' output $\mathbf{H}^t$, ground-truth $\mathbf{Y}$, previous performance on layer_{t-1}.
Output: threshold θ_t.
Procedure:
1: Initialize confidence set $\mathcal{S} = \emptyset$.
2: **if** Measure M is label-based **then**
3: **for** $j = 1$ **to** l **do**
4: compute confidence α_j^t on $\mathbf{H}_{\cdot j}^t$.
5: compute measure m_j^t on $(\mathbf{H}_{\cdot j}^t, \mathbf{Y}_{\cdot j})$.
6: $\mathcal{S} = \mathcal{S} \cup \{\alpha_j^t\}$ when m_j^t is worse than m_j^{t-1}.
7: **end for**
8: **end if**
9: **if** Measure M is instance-based **then**
10: **for** $i = 1$ **to** m **do**
11: compute confidence α_i^t on $\mathbf{H}_{i\cdot}^t$.
12: compute measure m_i^t on $(\mathbf{H}_{i\cdot}^t, \mathbf{Y}_{i\cdot})$.
13: $\mathcal{S} = \mathcal{S} \cup \{\alpha_i^t\}$ when m_i^t is worse than m_i^{t-1}.
14: **end for**
15: **end if**
16: θ_t = Compute threshold on $\mathcal{S}$.

$\alpha_j = \frac{1}{4}(0.9 + 0.6 + 0.6 + 0.7) = 0.7$. For one-error, we consider the probability that the most confident label is positive. For coverage, we sum the probability that the number of labels which can include covering all relevant labels and scale the result to range $[0, 1]$. For ranking loss, we compute the probability that the ranking loss is zero, which means the positive labels are ahead of negative labels. For example, if the prediction vector $\mathbf{p}_{i\cdot} = [0.9, 0.6, 0.4, 0.3]$, there will be 5 possible permutations of ground-truth leading to zero ranking loss: $\{0000, 1000, 1100, 1110, 1111\}$. The probability of each case is simple to obtain, e.g. $\Pr[1100] = 0.9 \cdot 0.6 \cdot (1 - 0.4) \cdot (1 - 0.3)$. Therefore, we can get confidence by summing the probabilities in these five cases. The confidence of macro-AUC and average precision is defined in a similar way.

Algorithm 1 summarizes the process of the measure-aware feature reuse. Due to the diversity between label-based measures and instance-based measures [25], we need to deal with them separately. Explicitly, the label-based measures compute the confidence of $\mathbf{H}^t$ on the third dimension, and the instance-based measures compute it on the second dimension. After the confidence computing, we reuse the previous representation $\mathbf{G}^{t-1}$ when the confidence α^t is below the threshold, and update $\mathbf{G}^t$ partially with the better ones.

The whole process of measure-aware feature reuse does not rely on true labels. We can judge the goodness of representation by a threshold determined in the training process. As Algorithm 2 shows, we save the confidence α^t into the set $\mathcal{S}$ when the evaluated performance measure goes worse at layer_t. Then, the threshold θ_t is determined based on the set $\mathcal{S}$, and we use the average of $\mathcal{S}$ as the threshold for convenience. Because the meaning of confidence is consistent with the measure, the threshold θ_t can be effectively utilized in the measure-aware feature reuse mechanism.

3.3 Measure-aware layer growth

The measure-aware feature reuse mechanism focuses on representation learning and can effectively enhance the representation guided by various measures. At the same time, to decrease overfitting and control the model complexity, we propose the measure-aware layer growth mechanism, which is used in the training process of MLDF.

Algorithm 3 Measure-aware layer growth

Input: maximal depth T, measure M, training data $\{\mathbf{X}, \mathbf{Y}\}$.
Output: model set $\mathcal{C}$, threshold set Θ and final layer index L.
Procedure:
1: Initialize parameters:
performance in each layer $\boldsymbol{q}[1:T]$,
best performance on train set q_{best},
the initial threshold $\theta_1 = 0$,
the best performance layer index $L = 1$,
the model set $\mathcal{C} = \emptyset$.
2: **for** $t = 1$ **to** T **do**
3: Train forests in layer_t and get classifier h_t.
4: Predict $\mathbf{H}_t = h_t([\mathbf{X}, \mathbf{G}_{t-1}])$.
5: $\theta_t =$ Determine Threshold (Algorithm 2) when $t > 1$.
6: $\mathbf{G}^t =$ measure-aware feature reuse (Algorithm 1).
7: Compute performance $\boldsymbol{q}[t]$ on measure M with $\mathbf{G}^t$.
8: **if** $\boldsymbol{q}[t]$ is better than q_{best} **then**
9: Update best performance $q_{\text{best}} = \boldsymbol{q}[t]$.
10: Update the layer index of best performance $L = t$.
11: **else if** q_{best} is not updated in recent 3 layers **then**
12: **break**
13: **end if**
14: Add layer_t to model set: $\mathcal{C} = \mathcal{C} \cup \text{layer}_t$.
15: **end for**
16: Keep $\{\text{layer}_1, \ldots, \text{layer}_L\}$ in model set $\mathcal{C}$ and drop others $\{\text{layer}_{L+1}, \ldots\}$.

If we use the same data to fit forests and do predict directly, the risk of overfitting will be increased [15]. MLDF uses the k-fold cross-validation to alleviate this issue. For each fold, we train the forests based on the training examples in other folds and predict the current fold. The layer's representation is generated by concatenating the predictions from each forest.

MLDF is built layer by layer. Algorithm 3 summarizes the procedure of measure-aware layer growth, which is the training process of MLDF. The inputs are maximal depth of layers T, evaluation measure M, and the training data $\{\mathbf{X}, \mathbf{Y}\}$. In general, we can choose one RF-PCT and one ERF-PCT in each layer and randomly select $\sqrt{d}$ number of features as candidates in each forest. All parameters of training forest in MLDF, such as the number of forests and the depth of trees, are pre-determined before training. As we hope each layer to learn different representations, we can set the maximum depth of tree in forests growing with the layer i, so does the number of trees, which can be set in advance. In the initialization step, the performance vector $\boldsymbol{q}$, which records the performance value on training data in each layer, should be initialized according to different measures. During each layer, we first fit the forests (Line 3) and get the representation $\mathbf{H}^t$ (Line 4). Then we should determine the threshold θ_t (Line 5) and generate new representation by measure-aware feature reuse (Line 6). Finally we add the layer_t to model set $\mathcal{C}$ (Line 14).

The layer growth is measure-aware. After fitting one layer, we are required to compute the evaluation measure. When the measure is not getting better in the recent three layers (Line 11), MLDF is forced to stop growing. At the same time, the layer index of the best performance in the training dataset should be recorded, which is useful for prediction. According to Occam's razor rule, we prefer a simpler model when the performance is similar. While there is no apparent improvement in performance, the final model set $\mathcal{C} = \{\text{layer}_1, \ldots, \text{layer}_L\}$ should be kept, and layers after layer_L will be dropped.

Different measures represent different user's demands. We can set M as the required measure according to various situations. Therefore, we can obtain the corresponding model for the specified measure. In summary, the measure-aware layer growth mechanism can control the model complexity and help the measure-aware feature reuse mechanism promote the performance.

4 EXPERIMENTS

In this section, we conduct experiments with MLDF on different multi-label classification benchmark datasets. Our goal is to validate that MLDF can achieve the best performance on different measures, and the two measure-aware mechanisms are necessary. Moreover, we show the advantages of MLDF through more detailed experiments from various aspects.

4.1 Dataset and configuration

We choose nine multi-label classification benchmark datasets from different application domains and with different scales. Table 3 presents the basic statistics of these datasets. All datasets are drawn from a repository of multi-label datasets[2]. The datasets vary in size: from 502 up to 43970 examples, from 68 up to 5000 features, and from 5 up to 201 labels. They are roughly organized in ascending order of the number of examples m, with eight of them being regular-scale, i.e., $m < 5000$ and eight of them being large-scale, i.e., $m > 5000$. For all experiments conducted below, 50% examples are randomly sampled without replacement to form the training set, and the rest 50% examples are used to create the test set.

Six evaluation measures, widely used in multi-label learning [25], are employed in this paper: hamming loss, one-error, coverage, ranking loss, average precision, and macro-AUC. Note that the coverage is normalized by the number of labels, and thus all the evaluation measures all vary between [0,1].

Hyper-parameters of MLDF are set as follows. We set the number of max layers (T) as 20 and take M with the six measures discussed above, respectively, which means we will get different models with different measures though other settings are the same. We take one RF-PCT and one ERF-PCT in each layer and use the 5-fold cross-validation to reduce overfitting. In the first layer, we take 40 trees in each forest, and then take 20 more trees than the previous layer until the number of trees reaches 100, which can enable MLDF to learn diverse representations. Similarly, we set the max-depth to 3 in the first layer, and then take three more than the previous layer when layer increasing.

Table 3. Descriptions of the datasets in terms of the domain (Domain), number of examples (m), number of features (d) and number of labels (l).

Dataset	Domain	m	d	l
CAL500	music	502	68	174
enron	text	1702	1001	53
image	images	2000	294	5
scene	images	2407	1196	6
yeast	biology	2417	103	14
corel16k-s1	images	13766	500	153
corel16k-s2	images	13761	500	164
eurlex-sm	text	19348	5000	201
mediamill	multimedia	43970	120	101

[2] http://mulan.sourceforge.net/datasets-mlc.html

4.2 Performance comparison

We compare MLDF to the following five contenders: a) RF-PCT [11], b) DBPNN [9, 19], c) MLFE [28], d) RAKEL [23] and e) ECC [18]. In the above, DBPNN is the representative of DNN methods; RAKEL, ECC, and RF-PCT are representatives of multi-label ensemble methods; MLFE is a method that utilizes the structural information in feature space to enrich the labeling information. Parameter settings of the compared methods are listed below. In detail, for RF-PCT, we take the amount as 100. For DBPNN, we conduct the experiments with Meka [19] and set the base classifier as the logistic function , other hyper-parameters are the same as recommended in Meka . For MLFE, we keep the same setting as suggested in [28], where $\rho = 1, c_1 = 1, c_2 = 2$, β_1, β_2 and β_3 are chosen among $\{1, 2, ..., 10\}, \{1, 10, 15\}$ and $\{1, 10\}$ respectively. The ensemble size of ECC is set to 100 to accommodate a sufficient number of classifier chains, and the ensemble size of RAKEL is set to $2q$ with $k = 3$ as suggested in the literature. The base learner of ECC and RAKEL is SVM with a linear kernel. For fairness, all methods use the 5-fold cross-validation.

We conduct experiments on each algorithm for ten times. The mean metric value and the standard deviation across ten training/testing trials are recorded for comparative studies. Table 4 reports the detailed experimental results of comparing algorithms. MLDF achieves optimal (lowest) average rank in terms of each evaluation measure. On the nine benchmark datasets, across all the evaluation measures, MLDF ranks 1st in 98.89% cases and ranks 2nd in 1.11% cases. Compared on these six measures, MLDF ranks 1st in 100.00%, 97.78%, 97.78%, 100.00%, 97.78%, 100.00% cases respectively. To summarize, MLDF achieves the best performance against other well-established contenders across extensive benchmark datasets on various evaluation measures, which validates the effectiveness of MLDF.

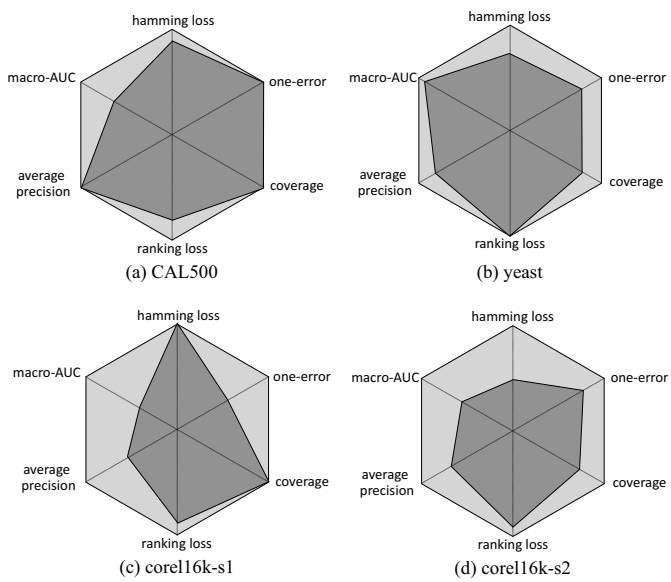

Figure 2. The performance comparison on CAL500, yeast, corel16k-s1 and corel16k-s2. The light hexagon represents the performance of MLDF, the darker represents the performance of MLDF without the measure-aware feature reuse mechanism. The larger the area means the better performance.

Table 4. Predictive performance (mean ± standard deviation) of each comparing methods on the nine datasets. •(○) indicates MLDF is significantly better(worse) than the comparing method on the criterion based on paired t-test at 95% significant level. ↓ (↑) means the smaller (larger) the value is, the performance will be the better.

Algorithm	CAL500	enron	image	scene	yeast	corel16k-s1	corel16k-s2	eurlex-sm	mediamill
hamming loss ↓									
MLDF	0.136±0.001	0.046±0.000	0.148±0.003	0.082±0.002	0.190±0.003	0.018±0.000	0.017±0.000	0.006±0.001	0.027±0.001
RF-PCT	0.137±0.001•	0.049±0.001•	0.156±0.002•	0.096±0.001•	0.196±0.002•	0.019±0.001•	0.018±0.001•	0.008±0.001•	0.028±0.001•
DBPNN	0.169±0.001•	0.075±0.001•	0.264±0.001•	0.260±0.001•	0.220±0.001•	0.029±0.001•	0.026±0.001•	0.007±0.001•	0.031±0.001•
MLFE	0.141±0.002•	0.047±0.001•	0.162±0.006•	0.084±0.002•	0.203±0.002•	0.019±0.001•	0.018±0.001•	0.007±0.001•	0.029±0.001•
ECC	0.182±0.005•	0.056±0.001•	0.218±0.027•	0.096±0.003•	0.207±0.003•	0.030±0.001•	0.018±0.001•	0.010±0.001•	0.035±0.001•
RAKEL	0.138±0.002•	0.058±0.001•	0.173±0.004•	0.096±0.004•	0.202±0.003•	0.020±0.001•	0.019±0.001•	0.007±0.001•	0.031±0.001•
one-error ↓									
MLDF	0.122±0.009	0.216±0.009	0.239±0.008	0.188±0.005	0.223±0.010	0.640±0.003	0.639±0.004	0.138±0.001	0.147±0.005
RF-PCT	0.122±0.010•	0.231±0.011•	0.258±0.005•	0.215±0.010•	0.247±0.008•	0.723±0.002•	0.721±0.006•	0.270±0.006•	0.150±0.002•
DBPNN	0.116±0.013○	0.490±0.012•	0.505±0.012•	0.690±0.003•	0.247±0.004•	0.740±0.004•	0.697±0.004•	0.460±0.015•	0.200±0.003•
MLFE	0.133±0.010•	0.232±0.003•	0.265±0.008•	0.201±0.005•	0.245±0.010•	0.680±0.005•	0.665±0.004•	0.345±0.010•	0.151±0.002•
ECC	0.137±0.021•	0.293±0.008•	0.408±0.069•	0.247±0.010•	0.244±0.009•	0.706±0.006•	0.712±0.005•	0.346±0.007•	0.150±0.005•
RAKEL	0.286±0.039•	0.412±0.016•	0.312±0.010•	0.247±0.009•	0.251±0.008•	0.886±0.007•	0.897±0.006•	0.447±0.016•	0.181±0.002•
coverage ↓									
MLDF	0.741±0.006	0.223±0.003	0.159±0.004	0.064±0.003	0.434±0.004	0.262±0.002	0.274±0.005	0.066±0.001	0.128±0.001
RF-PCT	0.756±0.007•	0.223±0.007•	0.170±0.004•	0.073±0.004•	0.436±0.007•	0.321±0.002•	0.310±0.002•	0.058±0.001○	0.133±0.001•
DBPNN	0.784±0.002•	0.292±0.006•	0.187±0.006•	0.084±0.004•	0.458±0.003•	0.370±0.002•	0.372±0.002•	0.552±0.011•	0.575±0.003•
MLFE	0.758±0.008•	0.237±0.007•	0.168±0.006•	0.080±0.008•	0.461±0.008•	0.368±0.002•	0.366±0.001•	0.085±0.002•	0.172±0.001•
ECC	0.806±0.016•	0.349±0.014•	0.229±0.034•	0.084±0.002•	0.464±0.005•	0.446±0.003•	0.436±0.002•	0.386±0.010•	0.467±0.009•
RAKEL	0.971±0.001•	0.523±0.008•	0.209±0.009•	0.104±0.003•	0.558±0.006•	0.667±0.002•	0.666±0.001•	0.543±0.012•	0.560±0.002•
ranking loss ↓									
MLDF	0.176±0.002	0.077±0.001	0.129±0.005	0.059±0.004	0.160±0.006	0.143±0.002	0.138±0.002	0.014±0.001	0.034±0.001
RF-PCT	0.178±0.002•	0.079±0.001•	0.142±0.004•	0.070±0.004•	0.164±0.008•	0.165±0.001•	0.142±0.001•	0.029±0.001•	0.035±0.001•
DBPNN	0.185±0.002•	0.126±0.007•	0.278±0.005•	0.277±0.005•	0.187±0.001•	0.154±0.002•	0.148±0.002•	0.396±0.011•	0.230±0.001•
MLFE	0.185±0.003•	0.082±0.008•	0.148±0.007•	0.065±0.004•	0.174±0.006•	0.189±0.002•	0.188±0.001•	0.034±0.002•	0.046±0.001•
ECC	0.204±0.008•	0.133±0.004•	0.224±0.043•	0.085±0.003•	0.186±0.003•	0.233±0.002•	0.229±0.001•	0.263±0.007•	0.179±0.008•
RAKEL	0.444±0.005•	0.241±0.005•	0.196±0.008•	0.107±0.003•	0.245±0.004•	0.414±0.002•	0.418±0.001•	0.388±0.011•	0.222±0.001•
average precision ↑									
MLDF	0.512±0.003	0.696±0.004	0.842±0.005	0.891±0.008	0.770±0.005	0.347±0.002	0.342±0.004	0.840±0.002	0.732±0.007
RF-PCT	0.512±0.006•	0.685±0.002•	0.829±0.003•	0.873±0.006•	0.758±0.008•	0.293±0.002•	0.287±0.002•	0.726±0.004•	0.729±0.001•
DBPNN	0.495±0.002•	0.500±0.007•	0.672±0.006•	0.563±0.004•	0.738±0.002•	0.289±0.002•	0.299±0.002•	0.427±0.013•	0.502±0.002•
MLFE	0.488±0.006•	0.688±0.009•	0.817±0.010•	0.882±0.005•	0.759±0.005•	0.319±0.001•	0.317±0.001•	0.853±0.007○	0.728±0.001•
ECC	0.482±0.008•	0.651±0.006•	0.739±0.043•	0.853±0.005•	0.752±0.006•	0.282±0.003•	0.276±0.002•	0.572±0.007•	0.597±0.014•
RAKEL	0.353±0.006•	0.539±0.006•	0.788±0.006•	0.843±0.005•	0.720±0.005•	0.103±0.003•	0.092±0.003•	0.440±0.013•	0.521±0.001•
macro-AUC ↑									
MLDF	0.568±0.006	0.742±0.014	0.885±0.003	0.956±0.003	0.732±0.010	0.728±0.001	0.737±0.007	0.930±0.002	0.842±0.002
RF-PCT	0.555±0.004•	0.729±0.012•	0.875±0.005•	0.947±0.002•	0.723±0.012•	0.712±0.004•	0.719±0.005•	0.904±0.007•	0.835±0.002•
DBPNN	0.499±0.001•	0.679±0.010•	0.746±0.006•	0.704±0.005•	0.627±0.004•	0.699±0.002•	0.708±0.003•	0.589±0.005•	0.510±0.001•
MLFE	0.547±0.006•	0.656±0.010•	0.841±0.006•	0.944±0.004•	0.705±0.005•	0.651±0.006•	0.662±0.002•	0.853±0.003•	0.799±0.002•
ECC	0.507±0.005•	0.646±0.008•	0.807±0.030•	0.931±0.004•	0.646±0.003•	0.627±0.004•	0.633±0.002•	0.624±0.004•	0.524±0.001•
RAKEL	0.547±0.007•	0.596±0.007•	0.803±0.005•	0.884±0.004•	0.614±0.003•	0.523±0.001•	0.525±0.001•	0.591±0.006•	0.513±0.001•

4.3 Influence of measure-aware feature reuse

The measure-aware feature reuse aims to reuse better representation in the previous layer according to confidence. When the confidence α^t is lower than threshold θ_t, we reuse the representation $\mathbf{G}^{t-1}$ in layer_{t-1}. If we skip the line 5 in Algorithm 3 and keep θ_t as 0 for all t in $[1, L]$, it is just that we do not take the measure-aware feature reuse mechanism in all layers. Figure 2 shows the comparison between using the mechanism and not using the mechanism on CAL500, yeast, corel16k-s1, and corel16k-s2. The six radii represent six different measures, the outermost part of the hexagon represents the performance of MLDF, the center represents the performance of RF-PCT, and the darker hexagon represents the performance of MLDF without the mechanism. The area of MLDF is larger than that of MLDF without the mechanism; therefore, it indicates that the measure-aware feature reuse mechanism is necessary on these datasets. Furthermore, the area of MLDF without the mechanism gets smaller when the size of datasets increases, which confirms that the measure-aware feature reuse mechanism does well in larger data.

4.4 Effect of measure-aware layer growth

We conduct experiments on the *yeast* dataset to show the effect of the measure-aware layer growth mechanism. Specifically, when MLDF arrives at its final layer index L, we keep the layer increase for observing whether the mechanism is sufficient. The number of trees in RF-PCT is 100, and the 5-fold cross-validation is used.

Figure 3 shows the *yeast*'s test performance curve of MLDF on six measures. The red circle means the final layer returned by Algorithm 3. For RF-PCT, we set the same trees as the final layer of MLDF (the red circle). The triangle indicates the performance of RF-PCT, which can be viewed as a one-layer MLDF. In Figure 3, the performance of MLDF becomes better when the model goes more in-depth, and our algorithm can stop almost at the position with the best performance. It demonstrates the effectiveness of our stopping mechanism. The performance of MLDF (the circle) is better than RF-PCT (the triangle), where they have the same amount of trees. Moreover, MLDF controlled by different measures can converge in different layers. It indicates that the measure-aware layer growth is sufficient.

Table 5. Predictive performance (mean ± standard deviation) of each comparing methods on all datasets. $(\cdot/\cdot/\cdot)$ indicates the times that MLDF based on RFML-C4.5 is significantly (superior/ equal/ inferior) to the comparing methods on the criterion based on paired t-test at 95% significant level.

dataset	hamming loss	one-error	coverage	ranking loss	average precision	macro-AUC
CAL500	0.137±0.001 (5/0/0)	0.120±0.018 (4/0/1)	0.738±0.004 (5/0/0)	0.176±0.002 (5/0/0)	0.511±0.005 (4/0/1)	0.569±0.007 (5/0/0)
enron	0.047±0.001 (5/0/0)	0.225±0.007 (5/0/0)	0.224±0.005 (4/0/1)	0.079±0.004 (5/0/0)	0.691±0.003 (5/0/0)	0.738±0.005 (5/0/0)
image	0.146±0.004 (5/0/0)	0.243±0.016 (5/0/0)	0.157±0.008 (5/0/0)	0.130±0.010 (5/0/0)	0.841±0.010 (5/0/0)	0.886±0.010 (5/0/0)
scene	0.083±0.003 (5/0/0)	0.192±0.007 (5/0/0)	0.063±0.002 (5/0/0)	0.059±0.002 (5/0/0)	0.889±0.004 (5/0/0)	0.956±0.003 (5/0/0)
yeast	0.190±0.003 (5/0/0)	0.225±0.009 (5/0/0)	0.434±0.006 (5/0/0)	0.159±0.002 (5/0/0)	0.769±0.003 (5/0/0)	0.733±0.006 (5/0/0)
corel16k-s1	0.019±0.001 (5/0/0)	0.726±0.002 (2/0/3)	0.316±0.012 (5/0/0)	0.163±0.007 (4/0/1)	0.295±0.002 (4/0/1)	0.695±0.002 (3/0/2)
corel16k-s2	0.018±0.001 (5/0/0)	0.727±0.001 (1/0/4)	0.313±0.016 (4/0/1)	0.160±0.009 (3/0/2)	0.284±0.007 (2/0/3)	0.697±0.011 (3/0/2)
eurlex-sm	0.007±0.001 (5/0/0)	0.201±0.012 (5/0/0)	0.043±0.001 (5/0/0)	0.021±0.001 (5/0/0)	0.784±0.008 (4/0/1)	0.900±0.002 (4/0/1)
mediamill	0.027±0.001 (5/0/0)	0.144±0.002 (5/0/0)	0.128±0.002 (5/0/0)	0.035±0.001 (5/0/0)	0.725±0.010 (3/0/2)	0.843±0.004 (5/0/0)
sum of score	45/0/0	37/0/8	43/0/2	42/0/3	37/0/8	40/0/5

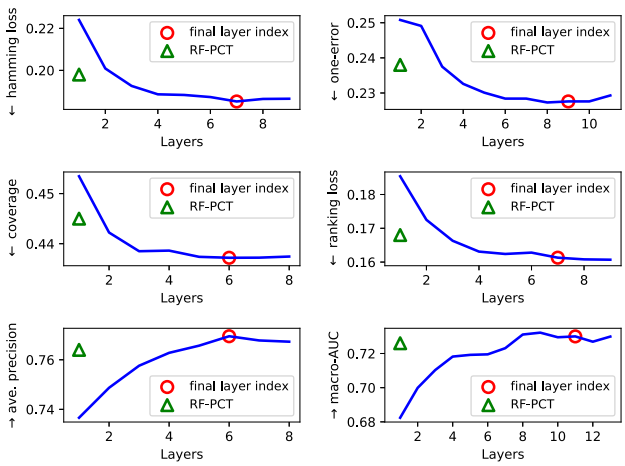

Figure 3. The test performance of MLDF in each layer on six measures on *yeast* dataset respectively. The triangle indicates the performance of RF-PCT, and the circle means the final layer index L. ↓ (↑) means the smaller (larger) the value is, the performance will be the better.

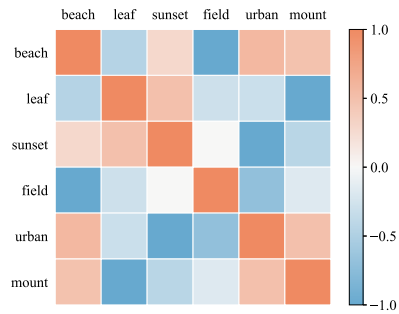

Figure 4. Effect of missing representation information on each label respectively. The *scene* dataset has 6 labels (top-to-down, left-to-right): "beach", "leaf", "sunset", "field", "urban" and "mountain".

4.5 Label correlations exploitation

Intuitively, the cascade structure enables MLDF to utilize label correlations. Thus we perform a distinctive approach to exploit label correlations. Our layer-wise method gradually considers more complex label correlations by utilizing the lower layer label representation in higher layer modeling. Here we deliberately delete a specific label in the first layer representation. Then train the second layer and check the influence of that malicious deletion. Suppose an accuracy decrease on label B is observed after deleting label A, we consider the two labels are correlated. The normalized relative decrease indicates the strength of correlations, and we show the result on *scene* dataset in Figure 4. As shown in Figure 4, label "beach" is highly correlated with label "urban" since sometimes they exist in *scene* dataset together [3]. It indicates that MLDF utilizes some correlations between labels to obtain better performance in inner layers.

4.6 Flexibility

In previous experiments, RF-PCT and ERF-PCT are the forest blocks in MLDF, which achieve the best performance. A natural question will be *is it possible to replace the forest block by other multi-label tree-based methods in MLDF, and how will the performance change*? To investigate this problem, we take one RFML-C4.5 and one ERFML-C4.5 in MLDF. To ensure fairness, we keep all the other configurations *same* as those in Section 4.2. Table 5 shows the result of MLDF based on RFML-C4.5. By comparing the results in Table 4, we count the times that MLDF wins/draws/loses the comparison. It is evident that MLDF based on RFML-C4.5 also achieves the best performance among all compared methods. Thus, no matter based on RF-PCT or RFML-C4.5, MLDF can achieve the best performance. It indicates that MLDF has excellent flexibility.

5 CONCLUSION

In this paper, we first introduce the deep forest framework to multi-label learning and propose Multi-Label Deep Forest (MLDF). The designed multi-layer structure enables MLDF to utilize correlations among labels. Because of the two measure-aware mechanisms, measure-aware feature reuse and measure-aware layer growth, our proposal can optimize different multi-label measures based on user's demand, reduce the risk of overfitting, and achieve the best results on a bunch of benchmark datasets. Experiments show that achieves excellent performance on wide-range benchmark datasets.

In the future, the efficiency of MLDF could be further improved by reusing some components during the process of forest training. We will try to find a way to interpret how high-order correlations use. Furthermore, we plan to embed extreme multi-label tree methods like FastXML [16] into MLDF and test the performance on extreme-scale multi-label problems.

ACKNOWLEDGEMENTS

This research was supported by the NSFC (61751306, 61673201). The authors would like to thank the anonymous reviewers for constructive suggestions, as well as Shen-Huan Lyu, Ming Pang and Peng Zhao for helpful discussions.

REFERENCES

[1] David Belanger, Bishan Yang, and Andrew McCallum, 'End-to-end learning for structured prediction energy networks', in *ICML*, pp. 429–439, (2017).

[2] Hendrik Blockeel, Luc De Raedt, and Jan Ramon, 'Top-down induction of clustering trees', in *ICML*, pp. 55–63, (1998).

[3] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown, 'Learning multi-label scene classification', *Pattern Recognition*, **37**(9), 1757–1771, (2004).

[4] Xiang Cheng, Shu-Guang Zhao, Xiao Xuan, and Kuo-Chen Chou, 'iatc-mhyb: a hybrid multi-label classifier for predicting the classification of anatomical therapeutic chemicals', *Oncotarget*, **8**(35), 58494, (2017).

[5] Amanda Clare and Ross D. King, 'Knowledge discovery in multi-label phenotype data', in *PKDD*, pp. 42–53, (2001).

[6] Ji Feng, Yang Yu, and Zhi-Hua Zhou, 'Multi-Layered Gradient Boosting Decision Trees', in *NeurIPS*, pp. 3555–3565, (2018).

[7] Ji Feng and Zhi-Hua Zhou, 'Autoencoder by forest', in *AAAI*, pp. 2967–2973, (2018).

[8] Pierre Geurts, Damien Ernst, and Louis Wehenkel, 'Extremely randomized trees', *Machine Learning*, **63**(1), 3–42, (2006).

[9] Geoffrey Hinton and Ruslan Salakhutdinov, 'Reducing the dimensionality of data with neural networks', *Science*, **313**(5786), 504–507, (2006).

[10] Dragi Kocev and Michelangelo Ceci, 'Ensembles of extremely randomized trees for multi-target regression', in *Discovery Science*, pp. 86–100, (2015).

[11] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski, 'Tree ensembles for predicting structured outputs', *Pattern Recognition*, **46**(3), 817–833, (2013).

[12] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski, 'An extensive experimental comparison of methods for multi-label learning', *Pattern Recognition*, **45**(9), 3084–3104, (2012).

[13] Jinseok Nam, Jungi Kim, Eneldo Loza Menc'ia, Iryna Gurevych, and Johannes Fürnkranz, 'Large-scale multi-label text classification - revisiting neural networks', in *ECML*, pp. 437–452, (2014).

[14] Ming Pang, Kai-Ming Ting, Peng Zhao, and Zhi-Hua Zhou, 'Improving deep forest by confidence screening', in *ICDM*, pp. 1194–1199, (2018).

[15] John C. Platt, 'Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods', in *Advances in Large Margin Classiers*, pp. 61–74. MIT Press, (1999).

[16] Yashoteja Prabhu and Manik Varma, 'Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning', in *KDD*, pp. 263–272, (2014).

[17] Jamie Ray, Heng Wang, Du Tran, Yufei Wang, Matt Feiszli, Lorenzo Torresani, and Manohar Paluri, 'Scenes-objects-actions: A multi-task, multi-label video dataset', in *ECCV*, pp. 660–676, (2018).

[18] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank, 'Classifier chains for multi-label classification', *Machine Learning*, **85**(3), 333–359, (2011).

[19] Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes, 'MEKA: A multi-label/multi-target extension to Weka', *Journal of Machine Learning Research*, **17**, 1–5, (2016).

[20] Robert E Schapire and Yoram Singer, 'Boostexter: A boosting-based system for text categorization', *Machine learning*, **39**(2-3), 135–168, (2000).

[21] Grigorios Tsoumakas and Ioannis Katakis, 'Multi-label classification: An overview', *International Journal of Data Warehousing and Mining*, **3**(3), 1–13, (2007).

[22] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas, 'Mining multi-label data', in *Data Mining and Knowledge Discovery Handbook*, 667–685, Springer, (2010).

[23] Grigorios Tsoumakas and Ioannis P. Vlahavas, 'Random k -labelsets: An ensemble method for multilabel classification', in *ECML*, pp. 406–417, (2007).

[24] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu, 'CNN-RNN: A unified framework for multi-label image classification', in *CVPR*, pp. 2285–2294, (2016).

[25] Xi-Zhu Wu and Zhi-Hua Zhou, 'A unified view of multi-label performance measures', in *ICML*, pp. 3780–3788, (2017).

[26] Min-Ling Zhang and Zhi-Hua Zhou, 'Multi-label neural networks with applications to functional genomics and text categorization', *IEEE Transactions on Knowledge and Data Engineering*, **18**(10), 1338–1351, (2006).

[27] Min-Ling Zhang and Zhi-Hua Zhou, 'A review on multi-label learning algorithms', *IEEE Transactions on Knowledge and Data Engineering*, **26**(8), 1819–1837, (2014).

[28] Qian-Wen Zhang, Yun Zhong, and Min-Ling Zhang, 'Feature-induced labeling information enrichment for multi-label learning', in *AAAI*, pp. 4446–4453, (2018).

[29] Ya-Lin Zhang, Jun Zhou, Wenhao Zheng, Ji Feng, Longfei Li, Ziqi Liu, Ming Li, Zhiqiang Zhang, Chaochao Chen, Xiaolong Li, and Zhi-Hua Zhou, 'Distributed deep forest and its application to automatic detection of cash-out fraud', *ACM Transactions on Intelligent Systems and Technology*, **10**(5), 1–19, (2019).

[30] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan, 'Deep semantic ranking based hashing for multi-label image retrieval', in *CVPR*, pp. 1556–1564, (2015).

[31] Peng Zhou and Nora El-Gohary, 'Ontology-based multilabel text classification of construction regulatory documents', *Journal of Computing in Civil Engineering*, **30**(4), 04015058, (2015).

[32] Zhi-Hua Zhou, *Ensemble Methods: Foundations and Algorithms*, Chapman and Hall/CRC, 2012.

[33] Zhi-Hua Zhou and Ji Feng, 'Deep forest: Towards an alternative to deep neural networks', in *IJCAI*, pp. 3553–3559, (2017).

[34] Zhi-Hua Zhou and Ji Feng, 'Deep Forest', *National Science Review*, **6**(1), 74–86, (2019).