Learning Gradient Boosted Multi-label Classification Rules

Chapter · February 2021

DOI: 10.1007/978-3-030-67664-3_8

Learning Gradient Boosted Multi-label Classification Rules

Michael Rapp (✉)[1], Eneldo Loza Mencía[1], Johannes Fürnkranz[2], Vu-Linh Nguyen[3], and Eyke Hüllermeier[3]

[1] Knowledge Engineering Group, TU Darmstadt, Germany
mrapp@ke.tu-darmstadt.de, research@eneldo.net
[2] Computational Data Analysis Group, JKU Linz, Austria
juffi@faw.jku.at
[3] Heinz Nixdorf Institute, Paderborn University, Germany
vu.linh.nguyen@uni-paderborn.de, eyke@upb.de

Abstract. In multi-label classification, where the evaluation of predictions is less straightforward than in single-label classification, various meaningful, though different, loss functions have been proposed. Ideally, the learning algorithm should be customizable towards a specific choice of the performance measure. Modern implementations of boosting, most prominently gradient boosted decision trees, appear to be appealing from this point of view. However, they are mostly limited to single-label classification, and hence not amenable to multi-label losses unless these are label-wise decomposable. In this work, we develop a generalization of the gradient boosting framework to multi-output problems and propose an algorithm for learning multi-label classification rules that is able to minimize decomposable as well as non-decomposable loss functions. Using the well-known Hamming loss and subset 0/1 loss as representatives, we analyze the abilities and limitations of our approach on synthetic data and evaluate its predictive performance on multi-label benchmarks.

Keywords: Multi-label classification · Gradient boosting · Rule learning

1 Introduction

Multi-label classification (MLC) is concerned with the per-instance prediction of a subset of relevant labels out of a predefined set of available labels. Examples of MLC include real-world applications like the assignment of keywords to documents, the identification of objects in images, and many more (see, e.g., [23] or [24] for an overview). To evaluate the predictive performance of multi-label classifiers, one needs to compare the predicted label set to a ground-truth set of labels. As this can be done in many different ways, a large variety of loss functions have been proposed in the literature, many of which are commonly used to compare MLC in experimental studies. As these measures may conflict with each other, optimizing for one loss function often leads to deterioration with respect to another loss. Consequently, an algorithm is usually not able to dominate its competitors on all measures [7].

For many MLC algorithms it is unclear what measure they optimize. On the other hand, there are also approaches that are specifically tailored to a certain loss function, such as the F1-measure [17], or the subset 0/1 loss [16]. *(Label-wise) decomposable* losses are particularly easy to minimize, because the prediction for individual labels can be optimized independently of each other [7]. For example, binary relevance (BR) learning transforms an MLC problem to a set of independent binary classification problems, one for each label. On the other hand, much of the research in MLC focuses on incorporating label dependencies into the prediction models, which is in general required for optimizing *non-decomposable* loss functions such as subset 0/1.

While BR is appropriate for optimizing the Hamming loss, another reduction technique, label powerset (LP), is a natural choice for optimizing the subset 0/1 loss [7]. The same applies to (probabilistic) classifier chains [4, 18], which seek to model the joint distribution of labels. Among the approaches specifically designed for MLC problems, boosting algorithms that allow for minimizing multi-label losses are most relevant for this work. Most of these algorithms (e.g. [25], [21], [13], [19] and variants thereof) require the loss function to be label-wise decomposable. This restriction also applies to methods that aim at minimizing ranking losses (e.g. [15], [5], or [19]) or transform the problem space to capture relations between labels (e.g. [14] or [2]). To our knowledge, AdaBoost.LC [1] is the only attempt at directly minimizing non-decomposable loss functions.

The first contribution of this work is a framework that allows for optimizing multivariate loss functions (Section 3). It inherits the advantages of modern formulations of the gradient boosting framework, in particular the ability to flexibly use different loss functions and to incorporate regularization into the learning objective. The proposed framework allows the use of non-decomposable loss functions and enables the ensemble members to provide loss-minimizing predictions for several labels at the same time, hence taking label dependencies into account. This is in contrast to AdaBoost.LC, where the base classifiers may only predict for individual labels. Our experiments suggest that this ability is crucial to effectively minimize non-decomposable loss functions on real-world data sets.

Our second contribution is BOOMER, a concrete instantiation of this framework for learning multi-label classification rules (Section 4). While there are several rule-based boosting approaches for conventional single-label classification, e.g., the ENDER framework [6] which generalizes several rule-based boosting methods such as RuleFit [9], their use has not yet been systematically investigated for MLC. We believe that rules are a natural choice in our framework, because they define a more general concept class than the commonly used decision trees: While each tree can trivially be viewed as a set of rules, not every rule set may be encoded as a tree. Also, an ensemble of rules provides more flexibility in how the attribute space is covered. While in an ensemble of T decision trees, each example is covered by exactly T rules, an ensemble of rules can distribute the rules in a more flexible way, using more rules in regions where predictions are difficult and fewer rules in regions that are easy to predict.

2 Preliminaries

In this section, we introduce the notation used throughout the remainder of this work and present the type of models we use. Furthermore, we present relevant loss functions and recapitulate to what extent their optimization may benefit from the exploitation of label dependencies.

In contrast to binary and multi-class classification, in multi-label classification an example can be associated with several class labels λ_k out of a predefined and finite label set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_K\}$. An example $\boldsymbol{x}$ is represented in attribute-value form, i.e., it consists of a vector $\boldsymbol{x} = (x_1, \ldots, x_L) \in \mathcal{X} = A_1 \times \cdots \times A_L$, where x_l specifies the value associated with a numeric or nominal attribute A_l. In addition, each example is associated with a binary label vector $\boldsymbol{y} = (y_1, \ldots, y_K) \in \mathcal{Y}$, where y_k indicates the absence (-1) or presence $(+1)$ of label λ_k. We denote the set of possible labelings by $\mathcal{Y} = \{-1, +1\}^K$.

We deal with MLC as a supervised learning problem in which the task is to learn a predictive model $f : \mathcal{X} \to \mathcal{Y}$ from a given set of labeled training examples $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} \subset \mathcal{X} \times \mathcal{Y}$. A model of this kind maps a given example to a predicted label vector $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})) \in \mathcal{Y}$. It should generalize well beyond the given observations, i.e., it should yield predictions that minimize the expected risk with respect to a specific loss function. In the following, we denote the binary label vector that is predicted by a multi-label classifier as $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_K) \in \mathcal{Y}$.

2.1 Ensembles of Additive Functions

We are concerned with ensembles $F = \{f_1, \ldots, f_T\}$ that consist of T additive classification functions $f_t \in \mathcal{F}$, referred to as *ensemble members*. By $\mathcal{F}$ we denote the set of potential classification functions. In this work, we focus on classification rules (cf. Section 2.2). Given an example $\boldsymbol{x}_n$, all of the ensemble members predict a vector of numerical confidence scores

$$\hat{\boldsymbol{p}}_n^t = f_t(\boldsymbol{x}_n) = \left(\hat{p}_{n1}^t, \ldots, \hat{p}_{nK}^t\right) \in \mathbb{R}^K, \tag{1}$$

where each score expresses a preference for predicting the label λ_k as absent if $\hat{p}_k < 0$ or as present if $\hat{p}_k > 0$. The scores provided by the individual members of an ensemble can be aggregated into a single vector of confidence scores by calculating the vector sum

$$\hat{\boldsymbol{p}}_n = F(\boldsymbol{x}_n) = \hat{\boldsymbol{p}}_n^1 + \cdots + \hat{\boldsymbol{p}}_n^T \in \mathbb{R}^K, \tag{2}$$

which can subsequently be turned into the final prediction of the ensemble in the form of a binary label vector (cf. Section 4.3).

2.2 Multi-label Classification Rules

As ensemble members, we use conjunctive classification rules of the form

$$f : b \to \hat{\boldsymbol{p}},$$

where b is referred to as the *body* of the rule and $\hat{\boldsymbol{p}}$ is called the *head*. The body $b : \mathcal{X} \to \{0, 1\}$ consists of a conjunction of conditions, each being concerned with one of the attributes. It evaluates to 1 if a given example satisfies all of the conditions, in which case it is said to be *covered* by the rule, or to 0 if at least one condition is not met. An individual condition compares the value of the l-th attribute of an example to a constant by using a relational operator, such as $=$ and $\neq$ (if the attribute A_l is nominal), or $\leq$ and $>$ (if A_l is numerical).

In accordance with (1), the head of a rule $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_K) \in \mathbb{R}^K$ assigns a numerical score to each label. If a given example $\boldsymbol{x}$ belongs to the axis-parallel region in the attribute space $\mathcal{X}$ that is covered by the rule, i.e., if it satisfies all conditions in the rule's body, the vector $\hat{\boldsymbol{p}}$ is predicted. If the example is not covered, a null vector is predicted. Thus, a rule can be considered as a mathematical function $f : \mathcal{X} \to \mathbb{R}^K$ defined as

$$f(\boldsymbol{x}) = b(\boldsymbol{x})\, \hat{\boldsymbol{p}}\,. \tag{3}$$

This is similar to the notation used by Dembczyński et al. [6] in the context of single-label classification. However, in the multi-label setting, we consider the head as a vector, rather than a scalar, to enable rules to predict for several labels.

2.3 Multi-label Loss Functions

Various measures for evaluating the predictions provided by a multi-label classifier are commonly used in the literature (see, e.g., [23] for an overview). We focus on measures that assess the quality of predictions for N examples and K labels in terms of a single score $\mathcal{L}(Y, \hat{Y}) \in \mathbb{R}_+$, where $Y, \hat{Y} \in \{-1, +1\}^{N \times K}$ are matrices that represent the true labels according to the ground truth, respectively the predicted labels provided by a classifier.

Selected Evaluation Measures The *Hamming loss* measures the fraction of incorrectly predicted labels among all labels and is defined as

$$\mathcal{L}_{\text{Hamm.}}\left(Y, \hat{Y}\right) \coloneqq \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} [\![y_{nk} \neq \hat{y}_{nk}]\!]\,, \tag{4}$$

where $[\![P]\!] = 1$ if the predicate P is true $= 0$ otherwise.

In addition, we use the *subset 0/1 loss* to measure the fraction of examples for which at least one label is predicted incorrectly. Is it formally defined as

$$\mathcal{L}_{\text{subs.}}\left(Y, \hat{Y}\right) \coloneqq \frac{1}{N} \sum_{n=1}^{N} [\![\boldsymbol{y}_n \neq \hat{\boldsymbol{y}}_n]\!]\,. \tag{5}$$

Both, the Hamming loss and the subset 0/1 loss, can be considered as generalizations of the 0/1 loss known from binary classification. Nevertheless, as will be discussed in the following Section 2.4, they have very different characteristics.

Surrogate Loss Functions As seen in Section 2.1, the members of an ensemble predict vectors of numerical confidence scores, rather than binary label vectors. For this reason, discrete functions, such as the bipartition measures introduced above, are not suited to assess the quality of potential ensemble members during training. Instead, continuous loss functions that can be minimized in place of the actual target measure ought to be used as surrogates. For this purpose, we use multivariate (instead of univariate) loss functions $\ell : \{-1, +1\}^K \times \mathbb{R}^K \to \mathbb{R}_+$, which take two vectors $\boldsymbol{y}_n$ and $\hat{\boldsymbol{p}}_n$ as arguments. The former represents the true labeling of an example $\boldsymbol{x}_n$, whereas the latter corresponds to the predictions of the ensemble members according to (2).

As surrogates for the Hamming loss and the subset 0/1 loss, we use different variants of the *logistic loss*. This loss function, which is equivalent to *cross-entropy*, is the basis for logistic regression and is commonly used in boosting approaches to single-label classification (early uses go back to Friedman et al. [8]). To cater for the characteristics of the Hamming loss, the *label-wise logistic loss* applies the logistic loss function to each label individually:

$$\ell_{\text{l.w.-log}}\left(\boldsymbol{y}_n, \hat{\boldsymbol{p}}_n\right) \coloneqq \sum_{k=1}^{K} \log\left(1 + \exp\left(-y_{nk}\hat{p}_{nk}\right)\right) . \tag{6}$$

Following the formulation of this objective, $\hat{p}_{nk}$ can be considered as log-odds, which estimates the probability of $y_{nk} = 1$ as $\text{logistic}\left(\hat{p}_{nk}\right) = \frac{1}{1+\exp(-\hat{p}_{nk})}$. Under the assumption of label independence, the logistic loss has been shown to be a consistent surrogate loss for the Hamming loss [5, 11].

As the label-wise logistic loss can be calculated by aggregating the values that result from applying the loss function to each label individually, it is label-wise decomposable. In contrast, the *example-wise logistic loss*

$$\ell_{\text{ex.w.-log}}\left(\boldsymbol{y}_n, \hat{\boldsymbol{p}}_n\right) \coloneqq \log\left(1 + \sum_{k=1}^{K} \exp\left(-y_{nk}\hat{p}_{nk}\right)\right) \tag{7}$$

is non-decomposable, as it cannot be computed via label-wise aggregation. This smooth and convex surrogate is proposed by Amit et al. [1], who show that it provides an upper bound of the subset 0/1 loss.

2.4 Label Dependence and (Non-)Decomposability

The idea of modeling correlations between labels to improve the predictive performance of multi-label classifiers has been a driving force for research in MLC for many years. However, Dembczyński et al. [7] brought up strong theoretical and empirical arguments that the type of loss function to be minimized, as well as the type of dependencies that occur in the data, strongly influence to what extent the exploitation of label dependencies may result in an improvement. The authors distinguish between two types of dependencies, namely *marginal (unconditional)* and *conditional dependence*. While the former refers to a lack of (stochastic) independence properties of the joint probability distribution $p\left(\boldsymbol{y}\right) = p\left(y_1, \ldots, y_K\right)$

on labelings $\boldsymbol{y}$, the latter concerns the conditional probabilities $p(\boldsymbol{y} \mid \boldsymbol{x})$, i.e., the distribution of labelings conditioned on an instance $\boldsymbol{x}$.

According to the notion given in Section 2.2, the rules we aim to learn contribute to the final prediction of labels for which they predict a non-zero confidence score and abstain otherwise. The head of a *multi-label rule* contains multiple non-zero scores, which enables one to express conditional dependencies between the corresponding labels, where the rule's body is tailored to cover a region of the attribute space where these dependencies hold. In contrast, *single-label rules* are tailored to exactly one label and ignore the others, for which reason they are unable to explicitly express conditional dependencies.

As discussed in Section 2.3, we are interested in the Hamming loss and the subset 0/1 loss, which are representatives for decomposable and non-decomposable loss functions, respectively. In the case of decomposability, modeling dependencies between the labels cannot be expected to drastically improve predictive performance [7]. For this reason, we expect that single-label rules suffice for minimizing Hamming loss on most data sets. In contrast, given that the labels in a data set are not conditionally independent, the ability to model dependencies is required to effectively minimize non-decomposable losses. Hence, we expect that the ability to learn multi-label rules is crucial for minimizing the subset 0/1 loss.

3 Gradient Boosting using Multivariate Loss Functions

As our first contribution, we formulate an extension of the gradient boosting framework to multivariate loss functions. This formulation, which should be flexible enough to use any decomposable or non-decomposable loss function, as long as it is differentiable, serves as the basis of the MLC method that is proposed in Section 4 as the main contribution of this paper.

3.1 Stagewise Additive Modeling

We aim at learning an ensemble of additive functions $F = \{f_1, \ldots, f_T\}$ as introduced in Section 2.1. It should be trained in a way such that the expected empirical risk with respect to a certain (surrogate) loss function ℓ is minimized. Thus, we are concerned with minimizing the regularized training objective

$$\mathcal{R}(F) = \sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{p}}_n) + \sum_{t=1}^{T} \Omega(f_t) \ , \tag{8}$$

where Ω denotes an (optional) regularization term that may be used to penalize the complexity of the individual ensemble members to avoid overfitting and to ensure the convergence towards a global optimum if ℓ is not convex.

Unfortunately, constructing an ensemble of additive functions that minimizes the objective given above is a hard optimization problem. In gradient boosting, this problem is tackled by training the model in a stagewise procedure, where the individual ensemble members are added one after the other, as originally

proposed by Friedman et al. [8]. At each iteration t, the vector of scores $F_t(\boldsymbol{x}_n)$ that is predicted by the existing ensemble members for an example $\boldsymbol{x}_n$ can be calculated based on the predictions of the previous iteration:

$$F_t(\boldsymbol{x}_n) = F_{t-1}(\boldsymbol{x}_n) + f_t(\boldsymbol{x}_n) = \left(\hat{\boldsymbol{p}}_n^1 + \cdots + \hat{\boldsymbol{p}}_n^{t-1}\right) + \hat{\boldsymbol{p}}_n^t \,. \tag{9}$$

Substituting the additive calculation of the predictions into the objective function given in (8) yields the following objective to be minimized by the ensemble member that is added in the t-th iteration:

$$\mathcal{R}(f_t) = \sum_{n=1}^{N} \ell\left(\boldsymbol{y}_n, F_{t-1}(\boldsymbol{x}_n) + \hat{\boldsymbol{p}}_n^t\right) + \Omega(f_t) \,. \tag{10}$$

3.2 Multivariate Taylor Approximation

To be able to efficiently minimize the training objective when adding a new ensemble member f_t, we rewrite (10) in terms of the second-order multivariate Taylor approximation

$$\mathcal{R}(f_t) \approx \sum_{n=1}^{N} \left(\ell\left(\boldsymbol{y}_n, F_{t-1}(\boldsymbol{x}_n)\right) + \boldsymbol{g}_n \hat{\boldsymbol{p}}_n^t + \frac{1}{2} \hat{\boldsymbol{p}}_n^t H_n \hat{\boldsymbol{p}}_n^t \right) + \Omega(f_t) \,, \tag{11}$$

where $\boldsymbol{g}_n = (g_{n1}, \ldots, g_{nK})$ denotes the vector of first-order partial derivatives of the loss function ℓ with respect to the existing ensemble members' predictions for a particular example $\boldsymbol{x}_n$ and individual labels λ_k. Accordingly, the Hessian matrix $H_n = \left(\left(h_{n11} \ldots h_{n1K}\right), \ldots, \left(h_{nK1} \ldots h_{nKK}\right)\right)$ consists of all second-order partial derivatives:

$$g_{ni} = \frac{\partial \ell}{\partial \hat{p}_{ni}}\left(\boldsymbol{y}_n, F_{t-1}(\boldsymbol{x}_n)\right), \quad h_{nij} = \frac{\partial \ell}{\partial \hat{p}_{ni} \partial \hat{p}_{nj}}\left(\boldsymbol{y}_n, F_{t-1}(\boldsymbol{x}_n)\right) \,. \tag{12}$$

By removing constant terms, (11) can be further simplified, resulting in the approximated training objective

$$\widetilde{\mathcal{R}}(f_t) = \sum_{n=1}^{N} \left(\boldsymbol{g}_n \hat{\boldsymbol{p}}_n^t + \frac{1}{2} \hat{\boldsymbol{p}}_n^t H_n \hat{\boldsymbol{p}}_n^t \right) + \Omega(f_t) \,. \tag{13}$$

In each training iteration, the objective function $\widetilde{\mathcal{R}}$ can be used as a quality measure to decide which of the potential ensemble members improves the current model the most. This requires the predictions of the potential ensemble members for examples $\boldsymbol{x}_n$ to be known. How to find these predictions depends on the type of ensemble members used and the loss function at hand. In Section 4, we present solutions to this problem, using classification rules as ensemble members.

Algorithm 1: Learning an ensemble of boosted classification rules

input : Training examples $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_n^N$, first and second derivative ℓ' and ℓ'' of the loss function, number of rules T, shrinkage parameter η
output: Ensemble of rules F

1 $\mathcal{G} = \{\boldsymbol{g}_n\}_n^N, \mathcal{H} = \{H_n\}_n^N$ = calculate gradients and Hessians w.r.t. ℓ' and ℓ''
2 $f_1 : b_1 \rightarrow \hat{\boldsymbol{p}}_1$ with $b_1(\boldsymbol{x}) = 1, \forall \boldsymbol{x}$ and $\hat{\boldsymbol{p}}_1 = \text{FIND\_HEAD}(\mathcal{D}, \mathcal{G}, \mathcal{H}, b_1)$ ▷ Section 4.1
3 **for** $t = 2$ **to** T **do**
4 $\quad \mathcal{G}, \mathcal{H}$ = update gradients and Hessians of examples covered by f_{t-1}
5 $\quad \mathcal{D}'$ = randomly draw N examples from $\mathcal{D}$ (with replacement)
6 $\quad f_t : b_t \rightarrow \hat{\boldsymbol{p}}_t = \text{REFINE\_RULE}(\mathcal{D}', \mathcal{G}, \mathcal{H})$ ▷ Section 4.2
7 $\quad \hat{\boldsymbol{p}}_t = \text{FIND\_HEAD}(\mathcal{D}, \mathcal{G}, \mathcal{H}, b_t)$
8 $\quad \hat{\boldsymbol{p}}_t = \eta \cdot \hat{\boldsymbol{p}}_t$
9 **return** ensemble of rules $F = \{f_1, \ldots, f_T\}$

4 Learning Boosted Multi-label Classification Rules

Based on the general framework defined in the previous section, we now present BOOMER, a concrete stagewise algorithm for learning an ensemble of gradient boosted single- or multi-label rules $F = \{f_1, \ldots, f_T\}$ that minimizes a given loss function in expectation[4]. The basic algorithm is outlined in Alg. 1.

Because rules, unlike other classifiers like e.g. decision trees, only provide predictions for examples they cover, the first rule f_1 in the ensemble is a *default rule*, which covers all examples, i.e., $b_1(\boldsymbol{x}) = 1, \forall \boldsymbol{x} \in \mathcal{X}$. In subsequent iterations, more specific rules are added. All rules, including the default rule, contribute to the final predictions of the ensemble according to their confidence scores, which are chosen such that the objective function in (13) is minimized. In each iteration t, this requires the gradients and Hessians to be (re-)calculated based on the confidence scores $\hat{p}_{nk}$ that are predicted for each example $\boldsymbol{x}_n$ and label λ_k by the current model ($\hat{p}_{nk} = 0, \forall n, k$ if $t = 1$), as well as the true labels $y_{nk} \in \{-1, +1\}$ (cf. Alg. 1, lines 1 and 4). While the default rule always provides a confidence score for each label, all of the remaining rules may either predict for a single label or for all labels, depending on a hyper-parameter. We consider both variants in the experimental study presented in Section 5. The computations necessary to obtain loss-minimizing predictions for the default rule and each of the remaining rules are presented in Section 4.1.

To learn the rules $f_2, \ldots, f_T$, we use a greedy procedure where the body is iteratively refined by adding new conditions and the head is adjusted accordingly in each step. The algorithm used for rule refinement is discussed in detail in Section 4.2. To reduce the variance of the ensemble members, each rule is learned on a different subsample of the training examples, randomly drawn with replacement as in *bagging*, which results in more diversified and less correlated rules. However, once a rule has been learned, we recompute its predictions on the entire training data (cf. Alg. 1, line 7), which we have found to effectively

new rules

[4] An implementation is available at `https://www.github.com/mrapp-ke/Boomer`

Algorithm 2: FIND_HEAD

input : (Sub-sample of) training examples $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_n^N$, gradients $\mathcal{G} = \{\boldsymbol{g}_n\}_n^N$, Hessians $\mathcal{H} = \{H_n\}_n^N$, body b
output: Single- or multi-label head $\hat{\boldsymbol{p}}$

1 $\boldsymbol{g} = \sum_n b(\boldsymbol{x}_n)\, \boldsymbol{g}_n, H = \sum_n b(\boldsymbol{x}_n)\, H_n$
2 **if** loss function is decomposable **or** searching for a single-label head **then**
3 $\hat{\boldsymbol{p}}$ = obtain $\hat{p}_k$ w.r.t. $\boldsymbol{g}$ and H for each label independently acc. to (17)
4 **if** searching for a single-label head **then**
5 $\hat{\boldsymbol{p}}$ = find best single-label prediction $\hat{p}_k \in \hat{\boldsymbol{p}}$ w.r.t. (13)
6 **else**
7 $\hat{\boldsymbol{p}}$ = obtain $(\hat{p}_1, \ldots, \hat{p}_K)$ w.r.t. $\boldsymbol{g}$ and H by solving the linear system in (16)
8 **return** head $\hat{\boldsymbol{p}}$

prevent overfitting the sub-sample used for learning the rule. As an additional measure to reduce the risk of fitting noise in the data, the scores predicted by a rule may be multiplied by a *shrinkage* parameter $\eta \in (0, 1]$ (cf. Alg. 1, line 8). Small values for η, which can be considered as the learning rate, reduce the impact of individual rules on the model [12].

4.1 Computation of Loss-minimizing Scores

As illustrated in Alg. 2, the function FIND_HEAD is used to find optimal confidence scores to be predicted by a particular rule f, i.e., scores that minimize the objective function $\widetilde{\mathcal{R}}$ introduced in (13). Because of the fact that rules provide the same predictions $\hat{\boldsymbol{p}}$ for all examples $\boldsymbol{x}_n$ they cover and abstain for the others, the objective function can be further simplified. As the addition is commutative, we can sum up the gradient vectors and Hessian matrices that correspond to the covered examples (cf. Alg. 2, line 1), resulting in the objective

$$\widetilde{\mathcal{R}}(f_t) = \boldsymbol{g}\hat{\boldsymbol{p}} + \frac{1}{2}\hat{\boldsymbol{p}} H \hat{\boldsymbol{p}} + \Omega(f_t) \,, \tag{14}$$

where $\boldsymbol{g} = \sum_n b(\boldsymbol{x}_n)\, \boldsymbol{g}_n$ denotes the element-wise sum of the gradient vectors and $H = \sum_n b(\boldsymbol{x}_n)\, H_n$ corresponds to the sum of the Hessian matrices.

To penalize extreme predictions, we use the L_2 regularization term

$$\Omega_{\text{L2}}(f_t) = \frac{1}{2}\lambda \left\| \hat{\boldsymbol{p}}^t \right\|_2^2 \,, \tag{15}$$

where $\|\boldsymbol{x}\|_2$ denotes the Euclidean norm and $\lambda \geq 0$ is the regularization weight.

To ensure that the predictions $\hat{\boldsymbol{p}}$ minimize the regularized training objective $\widetilde{\mathcal{R}}$, we equate the first partial derivative of (14) with respect to $\hat{\boldsymbol{p}}$ with zero:

$$\begin{aligned} \frac{\partial \widetilde{\mathcal{R}}}{\partial \hat{\boldsymbol{p}}}(f_t) =& \boldsymbol{g} + H\hat{\boldsymbol{p}} + \lambda\hat{\boldsymbol{p}} = \boldsymbol{g} + (H + \operatorname{diag}(\lambda))\,\hat{\boldsymbol{p}} = 0 \\ \iff& (H + \operatorname{diag}(\lambda))\,\hat{\boldsymbol{p}} = -\boldsymbol{g} \,, \end{aligned} \tag{16}$$

where diag (λ) is a diagonal matrix with λ on the diagonal.

(16) can be considered as a system of K linear equations, where $H + \text{diag}(\lambda)$ is a matrix of coefficients, $-\boldsymbol{g}$ is a vector of ordinates and $\hat{\boldsymbol{p}}$ is the vector of unknowns to be determined. For commonly used loss functions, including the ones in Section 2.3, the sums of Hessians h_{ij} and h_{ji} are equal. Consequently, the matrix of coefficients is symmetrical.

In the general case, i.e., if the loss function is non-decomposable, the linear system in (16) must be solved to determine the optimal multi-label head $\hat{\boldsymbol{p}}$. However, when dealing with a decomposable loss function, the first and second derivative with respect to a particular element $\hat{p}_i \in \hat{\boldsymbol{p}}$ is independent of any other element $\hat{p}_j \in \hat{\boldsymbol{p}}$. This causes the sums of Hessians h_{ij} that do not exclusively depend on $\hat{p}_i$, i.e., if $i \neq j$, to become zero. In such case, the linear system reduces to K independent equations, one for each label. This enables to compute the optimal prediction $\hat{p}_i$ for the i-th label as

$$\hat{p}_i = -\frac{g_i}{h_{ii} + \lambda}\,. \tag{17}$$

Similarly, when dealing with single-label rules that predict for the i-th label, the predictions $\hat{p}_j$ with $j \neq i$ are known to be zero, because the rule will abstain for the corresponding labels. Consequently, (17) can be used to determine the predictions of single-label rules even if the loss function is non-decomposable.

4.2 Refinement of Rules

To learn a new rule, we use a top-down greedy search, commonly used in inductive rule learning (see, e.g., [10] for an overview). Alg. 3 is meant to outline the general procedure and does not include any algorithmic optimizations that can drastically improve the computational efficiency in practice.

The search starts with an empty body that is successively refined by adding additional conditions. Adding conditions to its body causes the rule to become more specific and results in less examples being covered. The conditions, which may be used to refine an existing body, result from the values of the covered examples in case of nominal attributes or from averaging adjacent values in case of numerical attributes. In addition to bagging, we use *random feature selection*, as in random forests, to ensure that the rules in an ensemble are more diverse (cf. Alg. 3, line 2). For each condition that may be added to the current body at a particular iteration, the head of the rule is updated via the function FIND_HEAD that has already been discussed in Section 4.1 (cf. Alg. 3, line 6). As a restriction, in the case of single-label rules, each refinement of the current rule must predict for the same label (omitted in Alg. 3 for brevity). Among all refinements, the one that minimizes the regularized objective in (14) is chosen. If no refinement results in an improvement according to said objective, the refinement process stops. We do not use any additional stopping criteria.

Algorithm 3: REFINE_RULE

input : (Sub-sample of) training examples $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_n^N$, gradients $\mathcal{G} = \{\boldsymbol{g}_n\}_n^N$, Hessians $\mathcal{H} = \{H_n\}_n^N$, current rule f (optional)
output: Best rule f^*

$f^* = f$
$A' =$ randomly select $\lfloor \log_2(L-1) + 1 \rfloor$ out of L attributes from $\mathcal{D}$
foreach possible condition c on attributes A' and examples $\mathcal{D}$ **do**
  $f' : b' \rightarrow \hat{\boldsymbol{p}}' =$ copy of current rule f
  add condition c to body b'
  $\hat{\boldsymbol{p}}' =$ FIND_HEAD $(\mathcal{D}, \mathcal{G}, \mathcal{H}, b')$ ▷ Section 4.1
  if $\widetilde{\mathcal{R}}(f') < \widetilde{\mathcal{R}}(f^*)$ w.r.t. $\mathcal{G}$ and $\mathcal{H}$ **then**
    $f^* = f'$
if $f^* \neq f$ **then**
  $\mathcal{D}' =$ subset of $\mathcal{D}$ covered by f^*
  return REFINE_RULE $(\mathcal{D}', \mathcal{G}, \mathcal{H}, f^*)$
return best rule f^*

4.3 Prediction

Predicting for an example $\boldsymbol{x}_n$ involves two steps: First, the scores provided by individual rules are aggregated into a vector of confidence scores $\hat{\boldsymbol{p}}_n$ according to (2). Second, $\hat{\boldsymbol{p}}_n$ must be turned into a binary label vector $\hat{\boldsymbol{y}}_n \in \mathcal{Y}$. As it should minimize the expected risk with respect to the used loss function, i.e., $\hat{\boldsymbol{y}}_n = \arg\min_{\hat{\boldsymbol{y}}_n} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$, it should be chosen in a way that is tailored to the loss function at hand.

In case of the label-wise logistic loss in (6), we compute the prediction as

$$\hat{\boldsymbol{y}}_n = \left(\operatorname{sgn}(\hat{p}_{n1}), \ldots, \operatorname{sgn}(\hat{p}_{nK})\right) , \tag{18}$$

where $\operatorname{sgn}(x) = +1$ if $x > 0$ and $= -1$ otherwise.

To predict the label vector that minimizes the example-wise logistic loss given in (7), we return the label vector among the vectors in the training data, which minimizes the loss, i.e.,

$$\hat{\boldsymbol{y}}_n = \arg\min_{\boldsymbol{y} \in \mathcal{D}} \ell_{\text{ex.w.-log}}(\boldsymbol{y}, \hat{\boldsymbol{p}}_n) . \tag{19}$$

The main reason for picking only a known label vector was to ensure a fair comparison to the main competitor for non-decomposable losses, LP, which is also restricted to returning only label vectors seen in the training data. Moreover, Senge et al. [20] argue that often only a small fraction of the 2^K possible label subsets are observed in practice, for which reason the correctness of an unseen label combination becomes increasingly unlikely for large data sets, and, in fact, our preliminary experiments confirmed this.

5 Evaluation

We evaluate the ability of our approach to minimize Hamming and subset 0/1 loss, using the label-wise and example-wise logistic loss as surrogates. In each case, we consider both, single- and multi-label rules, resulting in four different variants (*l.w.-log. single*, *l.w.-log. multi*, *ex.w.-log. single*, and *ex.w.-log. multi*). For each of them, we tune the shrinkage parameter $\eta \in \{0.1, 0.3, 0.5\}$ and the regularization weight $\lambda \in \{0.0, 0.25, 1.0, 4.0, 16.0, 64.0\}$, using *Bootstrap Bias Corrected Cross Validation (BBC-CV)* [22], which allows to incorporate parameter tuning and evaluation in a computationally efficient manner.

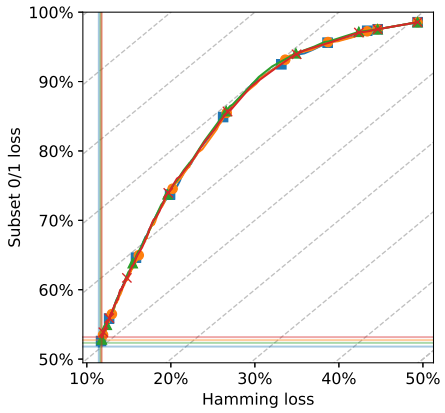
5.1 Synthetic Data

We use three synthetic data sets, each containing $10,000$ examples and six labels, as proposed by Dembczyński et al. [5] for analyzing the effects of using single- or multi-label rules with respect to different types of label dependencies. The attributes correspond to two-dimensional points drawn randomly from the unit circle and the labels are assigned according to linear decision boundaries through its origin. The results are shown in Fig. 1.

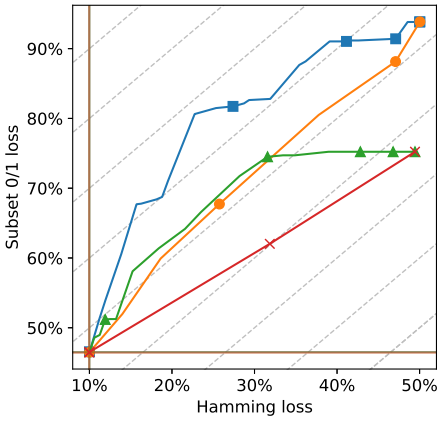
In the case of *marginal independence*, points are sampled for each label independently. Noise is introduced by randomly flipping 10% of the labels, such that the scores that are achievable by the Bayes-optimal classifier on an independent test set are 10% for Hamming loss and $\approx 47\%$ for subset 0/1 loss. As it is not possible — by construction of the data set — to find a rule body that is suited to predict for more than one label, multi-label rules cannot be expected to provide any advantage. In fact, despite very similar trajectories, the single-label variants achieve slightly better losses in the limit. This indicates that, most probably due to the number of uninformative features, the approaches that aim at learning multi-label rules struggle to induce pure single-label rules.

For modeling a *strong marginal dependence* between the labels, a small angle between the linear boundaries of two labels is chosen, so that they mostly coincide for the respective examples. Starting from different default rules, depending on the loss function they minimize, the algorithms converge to the same performances, which indicates that all variants can similarly deal with marginal dependencies. However, when using multi-label rules, the final subset 0/1 score is approached in a faster and more steady way, as a single rule provides predictions for several labels at once. In fact, it is remarkable that the ex.w.-log. multi variants already converge after two rules, whereas ex.w.-log. single needs six, one for each label, and optimizing for label-wise logarithmic loss takes much longer.

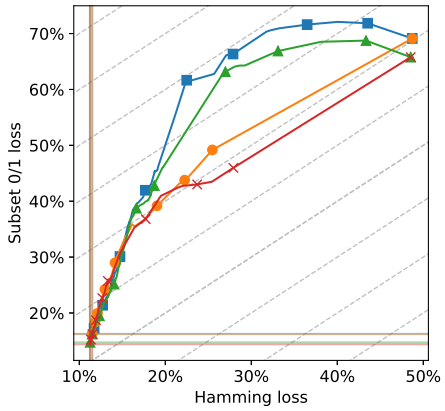
Finally, *strong conditional dependence* is obtained by randomly switching all labels for 10% of the examples. As a result, the score that is achievable by the Bayes-optimal classifier is 10% for both losses. While the trajectories are similar to before, the variants that optimize the non-decomposable loss achieve better results in the limit. Unlike the approaches that consider the labels independently, they seem to be less prone to the noise that is introduced at the example-level.



(a) Marginal independence



(b) Strong marginal dependence



(c) Strong conditional dependence

Fig. 1: Predictive performance of different approaches with respect to the Hamming loss and the subset 0/1 loss on three synthetic data sets. Starting at the top right, each curve shows the performance as more rules are added. The slope of the isometrics refers to an even improvement of Hamming and subset 0/1 loss. The symbols indicate $1, 2, 4, 8, \ldots, 512, 1000$ rules in the model.

5.2 Real-World Benchmark Data

We also conducted experiments on eight benchmark data sets from the Mulan and MEKA projects.[5] As baselines we considered binary relevance (BR), label powerset (LP) and classifier chains (CC) [18], using XGBoost [3] as the base classifier. For BR and CC we used the logistic loss, for LP we used the softmax objective. While we tuned the number of rules $T \in \{50, 100, \ldots, 10000\}$ for our algorithm, XGBoost comes with an integrated method for determining the number of trees. The learning rate and the L_2 regularization weight were tuned in the same value ranges for both. Moreover, we configured XGBoost to sample 66% of the training examples at each iteration and to choose from $\lfloor \log_2 (L-1) + 1 \rfloor$ random attributes at each split. As classifier chains are sensitive to the order of

[5] Data sets are available at `http://mulan.sourceforge.net/datasets-mlc.html` and `https://sourceforge.net/projects/meka/files/Datasets`.

Table 1: Predictive performance (in percent) of different approaches with respect to Hamming loss, subset 0/1 loss and the example-based F1-measure. For each evaluation measure and data set, we report the ranks of the different approaches (small numbers) and highlight the best approach (bold text).

		l.w.-log.		ex.w.-log.		XGBoost		
		single	multi	single	multi	BR	LP	CC
Subset 0/1 loss	BIRDS	38.72 2	40.43 6	39.57 4.5	39.15 3	39.57 4.5	40.85 7	**38.30** 1
	EMOTIONS	73.33 6.5	73.33 6.5	70.48 3.5	65.24 2	70.48 3.5	**60.00** 1	71.43 5
	ENRON	87.81 7	86.82 6	84.35 4	83.53 2	85.34 5	**82.70** 1	83.86 3
	LLOG	78.85 5	78.85 5	78.27 3	76.35 2	80.96 7	**70.38** 1	78.85 5
	MEDICAL	28.45 3	30.16 4	23.90 2	**23.04** 1	56.90 7	41.11 5	44.95 6
	SCENE	39.33 7	33.93 5	25.17 3	**23.26** 1	34.72 6	24.16 2	30.11 4
	SLASHDOT	65.29 5	62.89 4	53.30 3	**49.75** 1	72.04 7	52.94 2	66.96 6
	YEAST	83.72 7	82.27 5	78.60 4	75.81 2	82.72 6	**75.70** 1	76.59 3
	Avg. rank	5.31	5.19	3.38	**1.75**	5.75	2.50	4.13
Hamming loss	BIRDS	3.52 5	**3.16** 1	3.58 6	3.23 2	3.43 3	4.03 7	3.45 4
	EMOTIONS	18.81 5	20.00 7	18.17 2	18.41 3	18.73 4	**18.02** 1	19.29 6
	ENRON	4.56 3	**4.49** 1	4.90 5	4.91 6	4.52 2	5.75 7	4.63 4
	LLOG	1.48 2.5	1.48 2.5	1.49 4.5	**1.45** 1	1.49 4.5	2.13 7	1.60 6
	MEDICAL	0.84 2.5	0.86 4	0.84 2.5	**0.79** 1	1.69 7	1.55 6	1.36 5
	SCENE	8.16 7	7.42 6	7.21 4	**6.63** 1	7.19 3	7.27 5	6.85 2
	SLASHDOT	**4.15** 1	4.17 2	5.03 6	4.64 5	4.61 4	5.16 7	4.54 3
	YEAST	19.27 2	19.84 5	19.44 4	19.29 3	**19.13** 1	21.22 7	20.11 6
	Avg. rank	3.56	3.56	4.19	**2.75**	3.56	5.88	4.50
Example-based F1	BIRDS	70.38 4	**72.42** 1	68.00 7	71.18 2	69.68 6	70.06 5	70.96 3
	EMOTIONS	58.19 7	59.06 6	64.56 3	65.75 2	61.90 5	**69.24** 1	62.59 4
	ENRON	52.86 6	53.87 4	53.73 5	53.91 3	54.76 2	46.61 7	**56.46** 1
	LLOG	22.51 6	23.21 5	23.28 4	26.30 2	19.36 7	**33.17** 1	24.16 3
	MEDICAL	81.68 3	80.07 4	85.51 2	**85.97** 1	53.14 7	71.34 5	64.72 6
	SCENE	65.90 7	71.69 5	80.30 2	**81.69** 1	70.30 6	79.72 3	74.23 4
	SLASHDOT	40.94 5	44.04 4	54.78 2	**58.62** 1	32.57 7	54.01 3	39.15 6
	YEAST	61.56 6	61.03 7	62.78 3	**63.41** 1	62.54 4	61.71 5	62.92 2
	Avg. rank	5.50	4.50	3.50	**1.63**	5.50	3.75	3.63

labels, we chose the best order among ten random permutations. According to their respective objectives, the BR baseline is tuned with respect to Hamming, LP and CC are tuned with respect to subset 0/1 loss.

In Table 1, we report the predictive performance of our methods and their competitors in terms of Hamming and subset 0/1 loss. For completeness, we also report the example-based F1 score (see, e.g., [23]). The Friedman test indicates significant differences for all but the Hamming loss. The Nemenyi post-hoc test yields critical distances between the average ranks of 2.91/3.19 for $\alpha = 0.1/0.05$.

On average, *ex.w.-log. multi* ranks best in terms of subset 0/1 loss. It is followed by LP, its counterpart *ex.w.-log. single* and CC. As all of them aim at minimizing subset 0/1 loss, it is expected that they rank better than their competitors which aim at the Hamming loss. Most notably, since example-wise

optimized multi-label rules achieve better results than single-label rules on all data sets (statistically significant with $\alpha = 0.1$), we conclude that the ability to induce such rules, which is a novelty of the proposed method, is crucial for minimizing subset 0/1 loss.

On the other hand, in terms of Hamming loss, rules that minimize the label-wise logistic loss are competitive to the BR baseline, without a clear preference for single- or multi-label rules. Interestingly, although the example-wise logistic loss aims at minimizing subset 0/1 loss, when using multi-label rules, it also achieves remarkable results w.r.t. Hamming loss on some data sets and consequently even ranks best on average.

6 Conclusion

In this work, we presented an instantiation of the gradient boosting framework that supports the minimization of non-decomposable loss functions in multi-label classification. Building on this framework, we proposed an algorithm for learning ensembles of single- or multi-label classification rules. Our experiments confirm that it can successfully target different losses and is able to outperform conventional state-of-the-art boosting methods on data sets of moderate size.

While the use of multivariate loss functions in boosting has not received much attention so far, our framework could serve as a basis for developing algorithms specifically tailored to non-decomposable loss functions, such as F1 or Jaccard, and their surrogates. The main drawback is that the computation of predictions for a large number of labels n is computationally demanding in the non-decomposable case — solving the linear system has complexity $\mathcal{O}\left(n^3\right)$. To compensate for this, we plan to investigate approximations that exploit the sparsity in label space. As the training complexity in MLC not only increases with the number of examples and attributes, but also with the number of labels, such optimizations are generally required for handling very large data sets.

Acknowledgments This work was supported by the German Research Foundation (DFG) under grant number 400845550. Computations were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

References

1. Amit, Y., Dekel, O., Singer, Y.: A boosting algorithm for label covering in multilabel problems. In: Proc. Int. Conf. AI and Statistics (AISTATS). pp. 27–34 (2007)
2. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: Advances in Neural Information Processing Systems 28, pp. 730–738. Curran Associates, Inc. (2015)
3. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proc. 22nd Int. Conf. on Knowledge Discovery and Data Mining (KDD). p. 785–794 (2016)
4. Cheng, W., Hüllermeier, E., Dembczyński, K.: Bayes optimal multilabel classification via probabilistic classifier chains. In: Proc. 27th International Conference on Machine Learning (ICML). pp. 279–286 (2010)

5. Dembczyński, K., Kotłowski, W., Hüllermeier, E.: Consistent multilabel ranking through univariate losses. In: Proc. 29th Int. Conf. on Machine Learning (ICML). pp. 1319–1326. Omnipress (2012)
6. Dembczyński, K., Kotłowski, W., Słowiński, R.: ENDER: A statistical framework for boosting decision rules. Data Min. Knowl. Discov. **21**(1), 52–90 (2010)
7. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning **88**(1-2), 5–45 (2012)
8. Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics **28**(2), 337–407 (2000)
9. Friedman, J.H., Popescu, B.E.: Predictive learning via rule ensembles. Annals of Applied Statistics **2**, 916–954 (2008)
10. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer Science & Business Media (2012)
11. Gao, W., Zhou, Z.H.: On the consistency of multi-label learning. Artificial Intelligence **199-200**, 22–44 (2013)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media (2009)
13. Johnson, M., Cipolla, R.: Improved image annotation and labelling through multi-label boosting. In: Proc. British Machine Vision Conf. (BMVC) (2005)
14. Joly, A., Wehenkel, L., Geurts, P.: Gradient tree boosting with random output projections for multi-label classification and multi-output regression. arXiv preprint arXiv:1905.07558 (2019)
15. Jung, Y.H., Tewari, A.: Online boosting algorithms for multi-label ranking. In: Proc. 21st Int. Conf. on AI and Statistics (AISTATS). pp. 279–287 (2018)
16. Nam, J., Loza Mencía, E., Kim, H.J., Fürnkranz, J.: Maximizing subset accuracy with recurrent neural networks in multi-label classification. In: Advances in Neural Information Processing Systems 30 (NeurIPS). pp. 5419–5429 (2017)
17. Pillai, I., Fumera, G., Roli, F.: Designing multi-label classifiers that maximize F measures: State of the art. Pattern Recognition **61**, 394–404 (2017)
18. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: Proc. European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD). pp. 254–269. Springer (2009)
19. Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning **39**(2), 135–168 (2000)
20. Senge, R., del Coz, J.J., Hüllermeier, E.: Rectifying classifier chains for multi-label classification. In: Proceedings Lernen, Wissen & Adaptivität. pp. 151–158 (2013)
21. Si, S., Zhang, H., Keerthi, S.S., Mahajan, D., Dhillon, I.S., Hsieh, C.J.: Gradient boosted decision trees for high dimensional sparse output. In: Proc. 34th Int. Conf. on Machine Learning (ICML). pp. 3182–3190 (2017)
22. Tsamardinos, I., Greasidou, E., Borboudakis, G.: Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. Machine learning **107**(12), 1895–1922 (2018)
23. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer (2010)
24. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1819–1837 (2013)
25. Zhang, Z., Jung, C.: GBDT-MO: Gradient boosted decision trees for multiple outputs. arXiv preprint arXiv:1909.04373 (2019)