

Binary relevance efficacy for multilabel classification

Oscar Luaces · Jorge Díez · José Barranquero ·
Juan José del Coz · Antonio Bahamonde

Received: 17 September 2012 / Accepted: 29 September 2012 / Published online: 14 October 2012
© Springer-Verlag Berlin Heidelberg 2012

Abstract The goal of multilabel (ML) classification is to induce models able to tag objects with the labels that better describe them. The main baseline for ML classification is binary relevance (BR), which is commonly criticized in the literature because of its label independence assumption. Despite this fact, this paper discusses some interesting properties of BR, mainly that it produces optimal models for several ML loss functions. Additionally, we present an analytical study of ML benchmarks datasets and point out some shortcomings. As a result, this paper proposes the use of synthetic datasets to better analyze the behavior of ML methods in domains with different characteristics. To support this claim, we perform some experiments using synthetic data proving the competitive performance of BR with respect to a more complex method in difficult problems with many labels, a conclusion which was not stated by previous studies.

Keywords Multilabel classification · Binary relevance · Synthetic datasets · Label dependency

1 Introduction

Multilabel (ML) classification aims at obtaining models that provide a set of labels to each object, unlike multiclass classification that involves predicting just a single class. This learning task arises in many practical domains; for instance, media documents (texts, songs and videos) are usually tagged with several labels to briefly inform users about their actual content. Other well-known examples include

assigning keywords to a paper, illness to patients, objects to images or emotional expressions to human faces.

ML classification has received many contributions from different points of view. In [18] and [6], the approach was an extension of multiclass classification. Other methods follow different learning frameworks: this is the case of nearest neighbors [23], decision trees [21], Bayesian learners [1,22] and the combination of logistic regression and instance-based learning [3].

An interesting group of learners is based on the chain rule, using both inputs and labels in the learning process. In this group are [4,15] and [11]. Another successful approach consists in learning a ranking of labels for each instance and then, if necessary, producing a bipartition with a threshold that can be a fixed value or a variable learned from the learning task; see [6] and [13]. Finally, there are other approaches that aim to explicitly optimize a given loss function, for instance [4,5,12] and [13].

ML learning presents two main challenging problems. The first one bears on the computational complexity of the algorithms. If the number of labels is large, then a very complex approach is not applicable in practice, so the scalability is a key issue in this field. The second problem is related to the nature of ML data. Not only the number of classes is higher than in multiclass classification, but also each example belongs to an indeterminate number of labels, and more importantly, labels present some relationships between them. From a learning perspective, the hottest topic in ML community is probably to design new methods able to detect and exploit dependencies among labels. Actually, several methods have been proposed in that direction; for instance, those based on the chain rule cited previously.

Typically, all these new approaches are experimentally compared with *binary relevance* (BR) [7,19], the main baseline for ML classification. BR is a decomposition method

O. Luaces · J. Díez · J. Barranquero · J. J. del Coz (✉) · A. Bahamonde
Artificial Intelligence Center, University of Oviedo at Gijón,
Campus de Viesques, 33204 Gijón, Asturias, Spain
e-mail: juanjodelcoz@gmail.com
<http://www.aic.uniovi.es>

based on the learning assumption that labels are independent. Therefore, each label is classified as relevant or irrelevant by a binary classifier learned for that label independently from the rest of the labels. Despite that in those experimental studies BR is outperformed by new approaches, it should not be considered as a mere baseline. This paper proves that BR is not only computationally efficient, which is important in many practical situations, but is also *effective*, in the sense that it produces good ML classifiers according to several metrics. In fact, BR is able to induce optimal models when the target loss function is a macro-averaged measure. Moreover, this paper also supports the hypothesis that BR is competitive with respect to more complex approaches when the ML classification tasks are difficult, for instance, in domains with many labels and a high label dependency.

On the other hand, the experimental studies in ML classification have to deal with some issues. The most important one is the lack of rich collections of benchmark datasets. Although ML is a very active area of research, there are only few publicly available datasets. This fact conditions the experimental assessments of new proposals in such a way that some sophisticated algorithms seem to perform better than simpler learners, like BR, only because of the experimental sampling. Having in mind this context, our proposal is to combine benchmark and synthetic datasets to obtain empirical evidences about the actual performance of ML methods in different learning situations. Usually, synthetic data have been employed to illustrate a particular strength or weakness of a given algorithm, so in those cases the data generation process is algorithm specific. In contrast, we propose to build tools that can produce synthetic datasets useful to study the behavior of any new ML classification method.

In this paper, we employed a general-purpose generator of synthetic datasets for creating a collection of ML problems that reproduce a wide variety of situations. This generator¹ produces synthetic ML data allowing the user to select the desired values for some important characteristics, like the number of labels or the level of label dependency. Using a collection of synthetic datasets, we performed an exhaustive experiment comparing BR with a state-of-the-art method, ECC (*Ensemble of Classifier Chains*) [17]. The results of these experiments show some interesting evidences that BR is very competitive in hard ML problems.

The main contributions of this paper are threefold: (1) to formally discuss the properties of BR to obtain good models for macro-average loss functions; (2) to propose the use of synthetic datasets to remedy the shortcomings of benchmark datasets; (3) to experimentally prove the competitive performance of BR in ML domains in which the number of labels and the label dependency are high.

The rest of the paper is organized as follows. The next section gives a formal presentation of ML learning tasks and hypotheses. Section 3 reviews the main loss functions for ML classification. Then, we discuss the advantages of BR strategy, specially those related to its efficacy for optimizing macro-average loss functions. In Sect. 5, we present an analytical study about the properties of ML benchmark datasets. Next section describes a genetic algorithm to build ML synthetic datasets. Finally, Sect. 7 reports some experiments that support our claims and Sect. 8 draws some conclusions.

2 Multilabel classification

Let \mathcal{L} be a finite and non-empty set of labels $\{l_1, \dots, l_L\}$, let \mathcal{X} be an input space, and let \mathcal{Y} be the output space defined as the set of subsets of labels \mathcal{L} .

Definition 1 An *ML classification task* is given by a dataset

$$D = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\} \subset \mathcal{X} \times \mathcal{Y} \quad (1)$$

of pairs of inputs $\mathbf{x}_i \in \mathcal{X}$ and subsets of labels $\mathbf{y}_i \in \mathcal{Y}$ as outputs.

The labels assigned to each input are usually referred to as the *relevant* labels for the input entry.

Sometimes, when the input space is an Euclidean space of p dimensions, we will refer to the learning task by a couple of matrices

$$D \equiv (\mathbf{X}, \mathbf{Y}), \quad (2)$$

in which $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ and $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$. To make the notation clearer, each element of \mathbf{Y} , y_{ij} , is 1 when label j is relevant for example i , and 0 otherwise.

The goal of an ML classification task D is to induce a hypothesis defined as follows.

Definition 2 An *ML hypothesis* is a function h from the input space to the output space, the power set of labels $\mathcal{P}(\mathcal{L})$; in symbols,

$$h : \mathcal{X} \longrightarrow \mathcal{Y} = \mathcal{P}(\mathcal{L}) = \{0, 1\}^L. \quad (3)$$

Hence, $h(\mathbf{x})$ is the set of relevant labels predicted by h for the object \mathbf{x} . Sometimes, we use $h(\mathbf{X}) = \mathbf{Y}$ to mean that the predictions of h applied to an input set represented by a matrix \mathbf{X} are a set of labels codified by a matrix \mathbf{Y} .

3 Loss functions for multilabel classification

Multilabel classifiers can be evaluated from different points of view. The predictions can be considered either as a bipartition or as a ranking of the set of labels. This paper only studies ML classification tasks. Consequently, the performance

¹ It is available online at http://www.aic.uniovi.es/ml_generator.

of ML classifiers will be evaluated as a bipartition and the loss functions used must compare the subsets of relevant and predicted labels.

Usually, these kinds of measures can be divided into two main groups [20]. The *example-based* loss functions compute the average differences of the actual and the predicted sets of labels over all examples. The *label-based* measures decompose the evaluation with respect to each label. There are two options here, averaging the measure label-wise (usually called *macro-average*), or concatenating all label predictions and computing a single value over all of them, the *micro-average* version of a measure. Macro-average measures give equal weight to each label and are often dominated by the performance on rare labels. In contrast, micro-average metrics gives more weight to frequent labels. These two ways of measuring performance are complementary to each other and both are informative.

For further reference, let us recall the formal definitions of these loss functions, given an ML hypothesis h (3). For a prediction $h(\mathbf{x})$ and a subset of *truly relevant* labels $\mathbf{y} \subset \mathcal{L}$, for each label $l \in \mathcal{L}$ we can compute the contingency matrix in Table 1.

Each entry (a, b, c, d) in this matrix has a value of 1 when the predicates of the corresponding row and column are both true; otherwise, the value is 0. Notice, for instance, that $a(\mathbf{x}, l)$ is 1 only when the prediction of h includes the truly relevant label l . Furthermore, only one of the entries of the matrix is 1, while the rest are 0.

Throughout the definitions of the loss functions below, we will consider a set of n examples in an ML task with L labels. Additionally, we use the following aggregations of contingency matrices:

$$\begin{aligned}
 A_{x_i} &= \sum_{j=1}^L a(x_i, l_j) & B_{x_i} &= \sum_{j=1}^L b(x_i, l_j) \\
 C_{x_i} &= \sum_{j=1}^L c(x_i, l_j) & D_{x_i} &= \sum_{j=1}^L d(x_i, l_j) \\
 A_{l_j} &= \sum_{i=1}^n a(x_i, l_j) & B_{l_j} &= \sum_{i=1}^n b(x_i, l_j) \\
 C_{l_j} &= \sum_{i=1}^n c(x_i, l_j) & D_{l_j} &= \sum_{i=1}^n d(x_i, l_j) \\
 A &= \sum_{i=1}^n \sum_{j=1}^L a(x_i, l_j) & B &= \sum_{i=1}^n \sum_{j=1}^L b(x_i, l_j) \\
 C &= \sum_{i=1}^n \sum_{j=1}^L c(x_i, l_j) & D &= \sum_{i=1}^n \sum_{j=1}^L d(x_i, l_j)
 \end{aligned}$$

Definition 3 The *Recall* is defined as the proportion of truly relevant labels that are included in predictions. The example-based, macro- and micro-average versions are computed as follows:

Table 1 Contingency matrix for each label $l \in \mathcal{L}$ given the actual relevant, \mathbf{y} , and the predicted, $h(\mathbf{x})$, labels

	$\mathbf{y}[l] = 1$	$\mathbf{y}[l] = 0$
$h(\mathbf{x})[l] = 1$	$a(\mathbf{x}, l)$	$b(\mathbf{x}, l)$
$h(\mathbf{x})[l] = 0$	$c(\mathbf{x}, l)$	$d(\mathbf{x}, l)$

$$\begin{aligned}
 R^{\text{ex}} &= \frac{1}{n} \sum_{i=1}^n \frac{A_{x_i}}{A_{x_i} + C_{x_i}}, \\
 R^{\text{ma}} &= \frac{1}{L} \sum_{j=1}^L \frac{A_{l_j}}{A_{l_j} + C_{l_j}}, \\
 R^{\text{mi}} &= \frac{A}{A + C}.
 \end{aligned}$$

Definition 4 The *Precision* is defined as the proportion of predicted labels that are truly relevant. Example-based, macro and micro versions are defined by

$$\begin{aligned}
 P^{\text{ex}} &= \frac{1}{n} \sum_{i=1}^n \frac{A_{x_i}}{A_{x_i} + B_{x_i}}, \\
 P^{\text{ma}} &= \frac{1}{L} \sum_{j=1}^L \frac{A_{l_j}}{A_{l_j} + B_{l_j}}, \\
 P^{\text{mi}} &= \frac{A}{A + B}.
 \end{aligned}$$

The trade-off between *Precision* and *Recall* is formalized by their *harmonic* mean, called *F-measure*. $F_\beta (\beta \geq 0)$ is computed by

$$F_\beta = \frac{(1 + \beta^2)P \cdot R}{\beta^2 P + R}$$

Definition 5 Example-based, macro and micro $F_\beta (\beta \geq 0)$ are defined by

$$\begin{aligned}
 F_\beta^{\text{ex}} &= \frac{1}{n} \sum_{i=1}^n \frac{(1 + \beta^2)A_{x_i}}{(1 + \beta^2)A_{x_i} + B_{x_i} + \beta^2 C_{x_i}}, \\
 F_\beta^{\text{ma}} &= \frac{1}{L} \sum_{j=1}^L \frac{(1 + \beta^2)A_{l_j}}{(1 + \beta^2)A_{l_j} + B_{l_j} + \beta^2 C_{l_j}}, \\
 F_\beta^{\text{mi}} &= \frac{(1 + \beta^2)A}{(1 + \beta^2)A + B + \beta^2 C}.
 \end{aligned} \tag{4}$$

F_1 is the most frequently used *F-measure*.

Other performance measures for ML classifiers can also be defined using contingency matrices (Table 1). This is the case for the *Accuracy* and *Hamming loss*.

Definition 6 The *Accuracy* [19], or the *Jaccard index*, is a slight modification of the F_1 measure defined as

$$A_{c^{ex}} = \frac{1}{n} \sum_{i=1}^n \frac{A_{x_i}}{A_{x_i} + B_{x_i} + C_{x_i}},$$

$$A_{c^{ma}} = \frac{1}{L} \sum_{j=1}^L \frac{A_{l_j}}{A_{l_j} + B_{l_j} + C_{l_j}},$$

$$A_{c^{mi}} = \frac{A}{A + B + C}.$$

Definition 7 The *Hamming loss* is the proportion of misclassifications. The macro-average is given by

$$HI^{ma} = \frac{1}{L} \sum_{j=1}^L \frac{B_{l_j} + C_{l_j}}{A_{l_j} + B_{l_j} + C_{l_j} + D_{l_j}}. \quad (5)$$

Taking into account that the sum of the components of contingency matrices (see Table 1) is 1, the macro Hamming loss can be written as

$$HI^{ma} = \frac{1}{L} \sum_{j=1}^L \frac{B_{l_j} + C_{l_j}}{n} = \frac{B + C}{L \cdot n}$$

$$= HI^{mi}.$$

Moreover,

$$HI^{ma} = HI^{mi}$$

$$= \frac{1}{n} \sum_{i=1}^n \frac{B_{x_i} + C_{x_i}}{L}$$

$$= HI^{ex}.$$

That is to say, the Hamming loss is a measure that has the same value in their macro, micro and example-based versions.

Finally, another important ML example-based metric is the *Subset zero-one loss*.

Definition 8 The *Subset zero-one loss* examines if the predicted and relevant label subsets are equal or not:

$$S_{0/1} = \frac{1}{n} \sum_{i=1}^n \llbracket y_i \neq h(x_i) \rrbracket. \quad (6)$$

in which the expression $\llbracket p \rrbracket$ evaluates to 1 if the predicate p is true, and to 0 otherwise. This metric is an extension of the classical zero-one loss in multiclass classification to the ML case.

4 Binary relevance: a not so simple baseline

Binary relevance is a straightforward approach to handle an ML classification task. In fact, BR is usually employed as the baseline method to be compared with new ML methods. It is the simplest strategy, but is more effective than it may seem at first sight.

BR decomposes the learning of h into a set of binary classification tasks, one per label, where each single model h_j is learned independently, using only the information of that particular label and ignoring the information of all other labels. In symbols,

$$h_j : \mathcal{X} \longrightarrow \{0, 1\}.$$

The main drawback of BR is that it does not take into account any label dependency and may fail to predict some label combinations if such dependence is present. However, BR presents several obvious advantages: (1) any binary learning method can be taken as base learner; (2) it has linear complexity with respect to the number of labels; and (3) it can be easily parallelized. But the most important advantage of BR is that it is able to optimize *several* loss functions.

Given an ML classification task D (1), let M be a performance measure defined for a pair of lists of subsets of labels:

$$M((y_1, \dots, y_n), (\hat{y}_1, \dots, \hat{y}_n)) = M(\mathbf{Y}, \hat{\mathbf{Y}}),$$

where \mathbf{Y} and $\hat{\mathbf{Y}}$ represent the lists of subsets of actual and predicted labels, respectively. If higher values of M are preferable to lower, then the optimal predictions for the list of inputs \mathbf{X} are given by a hypothesis

$$h_M^*(\mathbf{X}) = \arg \max_{\hat{\mathbf{Y}}} \sum_{\mathbf{Y}} \Pr(\mathbf{Y}|\mathbf{X}) \cdot M(\mathbf{Y}, \hat{\mathbf{Y}}). \quad (7)$$

It is straightforward to see that the optimization of macro-averaged measures is equivalent to the optimization of those measures in the subordinate BR classifiers. When M is a macro-averaged measure, the optimization of (7) can be decomposed through the set of labels. Thus,

$$h_M^*(\mathbf{X}) = \arg \max_{\hat{\mathbf{Y}}} \sum_{\mathbf{Y}} \Pr(\mathbf{Y}|\mathbf{X}) \frac{1}{L} \left(\sum_{j=1}^L M(\mathbf{Y}[j], \hat{\mathbf{Y}}[j]) \right)$$

$$= \arg \max_{\hat{\mathbf{Y}}} \sum_{\mathbf{Y}} \sum_{j=1}^L \Pr(\mathbf{Y}|\mathbf{X}) M(\mathbf{Y}[j], \hat{\mathbf{Y}}[j]), \quad (8)$$

in which $\mathbf{Y}[j]$ is the j th column of matrix \mathbf{Y} that represents the corresponding label, l_j . Notice that this equation holds for all macro-average measures defined in Sect. 3, including Hamming loss in any of its versions since they are equal.

The consequence of (8) is that optimal predictions can be built from optimal outputs for each label for classification tasks drawn from the same distribution of the original ML task. That is, optimal BR classifiers will yield the optimal predictions.

Proposition 1 (Macro-average optimization) *The optimization of a macro-averaged measure M for an ML task can be accomplished by the optimization of the subordinate BR classifiers for the binary version of M .*

Table 2 Description of MULAN datasets

Dataset	ex.	att.	lab.	card.	dep.	den. (%)
bookmarks	87,856	2,150	208	2.03	0.10	0.98
CAL500	502	68	174	26.04	0.14	14.97
bibtex	7,395	1,836	159	2.40	0.15	1.51
bow	43,907	100	101	4.38	0.22	4.33
mpeg	43,907	320	101	4.38	0.22	4.33
Corel5k	5,000	499	374	3.52	0.15	0.94
Corel16k	13,766	500	153	2.86	0.14	1.87
delicious	16,105	500	983	19.02	0.11	1.93
emotions	593	72	6	1.87	0.38	31.14
enron	1,702	1,001	53	3.38	0.12	6.37
eurlex-dc	19,348	5,000	412	1.29	0.21	0.31
eurlex-ev	19,348	1,208	3,993	5.31	0.11	0.13
eurlex-sm	19,348	8,792	201	2.21	0.19	1.10
genbase	662	1,186	27	1.25	0.54	4.64
mediamill	43,907	120	101	4.38	0.22	4.33
medical	978	1,449	45	1.25	0.18	2.77
rcv1	6,000	47,236	101	2.88	0.21	2.85
scene	2,407	294	6	1.08	0.11	17.90
tmc2007	28,596	49,060	22	2.16	0.10	9.81
yeast	2,417	103	14	4.24	0.25	30.26
Median	10,580	500	101	2.87	0.17	3.59

For each dataset, the table shows the number of examples, the number of attributes, the number of labels, and the values for the cardinality (9), unconditional label dependency (11) and density (10). There are ten different versions of Corel16k dataset in the MULAN repository. We have only included one of them in this study because they have similar properties. The same happens with the rcv1 datasets, with five distinct subsets. Tmc2007 dataset has another version with only 500 attributes, but again the rest of the properties are practically the same

One consequence of this result affects the optimization of the Hamming loss, since it can be seen as the macro-average of the binary error rates of the labels (5).

Corollary 1 (Hamming loss optimization) *The optimization of Hamming loss for an ML task can be accomplished by the optimization of the subordinate BR classifiers for the binary error rate.*

The same corollary can be stated for all other macro-average loss functions. To optimize such measures, the binary classifiers that compose a BR model must optimize the corresponding binary measure. For instance, the optimization of macro F_1 requires that the binary classifiers optimize F_1 , using algorithms like the one proposed by [8].

This section proves that BR is not just a baseline classifier, but provides optimal models for several loss functions. For this reason, when new proposed ML learners are compared with BR using macro-averaged measures, the comparison must be done carefully, otherwise the derived conclusions may be biased. Another consequence is that future research in the field of ML classification should be focused on obtaining new theoretically sound methods able to optimize other kinds of loss functions, like example-based or micro-average measures.

5 Multilabel benchmark datasets

Despite the misuse of BR in comparative studies, there is another important issue when ML methods are analyzed empirically. The key problem is that there are just a few publicly available ML datasets. The most popular repository is maintained in the MULAN website. MULAN [20] is a WEKA extension for ML. Table 2 reports the main properties of MULAN’s datasets.

In addition to the scarcity of datasets, it is surprising that most of them are almost multiclass learning tasks. This can be measured using the *cardinality*: that is, the average number of labels per example. In symbols, the cardinality of a dataset is given by

$$\text{cardinality}(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^n \sum_{j=1}^L y_{i,j}}{n}. \tag{9}$$

In the datasets of the MULAN repository, the cardinality is very low; the median is just 2.87. Moreover, only 3 datasets out of 20 have a cardinality >5 , and 11 datasets have a cardinality lower than 3.

One important consequence of this fact is that the proportion of ones in the matrix \mathbf{Y} of labels is also very low. This proportion is called the *density* of the dataset and it is defined

as the cardinality divided by the number of possible labels,

$$\text{density}(\mathbf{X}, \mathbf{Y}) = \frac{\text{cardinality}(\mathbf{X}, \mathbf{Y})}{L}. \quad (10)$$

The median of the density in MULAN's datasets is only 3.59% (expressed as a percentage). Therefore, in some datasets a hypothesis predicting no labels for any input will have a very low percentage of misclassifications. This is a good reason to carefully consider if the Hamming loss is as an appropriate performance measure for a given domain.

Nevertheless, the key ingredient that makes ML an interesting research problem is that the labels show some kind of *dependency* between them. Otherwise, if the label independence assumption was fulfilled, BR would be the perfect approach. Thus, we need datasets with different levels of label dependency to evaluate the behavior of ML methods. Unfortunately, how to measure label dependency is not trivial.

From a Bayesian point of view, there are two possible kinds of dependencies: the conditional and the unconditional dependency [1, 4, 9, 22]. There are *conditional* dependencies between labels whenever

$$\Pr(l_1, \dots, l_L | \mathbf{X}) \neq \prod_{j=1}^L \Pr(l_j | \mathbf{X}).$$

This means that there are disjoint subsets of labels such that

$$\Pr((l_j : j \in J) | \mathbf{X}) \neq \Pr((l_j : j \in J) | \mathbf{X}, (l_i : i \in I)).$$

On the other hand, the dependency between labels, if there is any, is *unconditional* when the reference to input variables of the above equations can be skipped. In this paper we measure this kind of dependency as the average of the correlation of labels weighted by the number of common examples. In symbols,

$$\text{dependency}(\mathbf{Y}) = \frac{\sum_{i < j} \rho(l_i, l_j) |l_i \cap l_j|}{\sum_{i < j} |l_i \cap l_j|}, \quad (11)$$

in which $\rho(l_i, l_j)$ represents the absolute value of the correlation coefficient between labels l_i and l_j .

Looking at the values of unconditional label dependency in Table 2, we see that half of the datasets have a label dependency in the range [0.1...0.15]. Only two datasets have a value greater than 0.25. It is quite evident that the distribution of label dependency in MULAN's datasets is not very diverse. Viewing these numbers, how can one argue that a particular method is able to exploit label dependency if the experiments were performed using these datasets?

This brief analysis shows that the current collection of benchmark datasets presents important limitations, especially because ML problems are much more complex than those of other learning tasks, due to their own characteristics. For instance, in comparison with binary or multiclass

classification, ML classification has additional factors that are crucial, mainly the cardinality and the label dependency. This fact suggests that to study ML approaches experimentally, we should need more datasets than for the same kind of experiment in the context of multiclass classification.

Our statement is that benchmark datasets do not provide enough support for the experimental study of ML methods. For this reason, we propose to use them in combination with collections of synthetic datasets specially devised to offer a wider range of characteristics. In the next subsection, we describe one method that can be used to generate these collections.

6 A generator of synthetic datasets

Strictly speaking, there are no generators of synthetic ML learning tasks published. The approach presented in [14, 16] is mainly concerned with streaming data and can hardly be used to obtain ML datasets with a realistic combination of the properties described in the previous section.

Generating an ML dataset is not a trivial task. If one tries to concatenate several binary classification tasks with the same input instances, the result is that the labels will have no relationship at all. But, as stated before, it is mandatory to obtain some kind of dependency among labels.

In this paper, we used a genetic algorithm² to search for ML datasets with a set of target characteristics selected by the user. The goal is to obtain, for each desired combination of properties, three datasets: train, validation and test, with approximately the same property values. They will be represented by matrices as in (2).

6.1 Data generation

In all cases, the input space will be a hypercube

$$\mathcal{X} = [0, 1]^p \subset \mathbf{R}^p. \quad (12)$$

The requisites of the search set by the user include: number of labels, number of examples for test, training and validation sets, cardinality and dependency.

All parameters but cardinality and dependency are somehow structural and can be easily fulfilled. Thus, the generator starts with a set of inputs drawn from a uniform distribution in \mathcal{X} ; let \mathbf{X} be the matrix of input instances for the training set. The core idea of the generator presented here is that it searches for a *hypothesis* to classify the inputs in \mathbf{X} and obtain an ML task with cardinality and dependency as close as possible to those specified by the user.

Once the hypothesis is found, the validation and test datasets are built; their input instances are independently

² Website: http://www.aic.uniovi.es/ml_generator.

drawn with uniform distribution again, from the input space \mathcal{X} . This guarantees that training, validation and testing examples come from the same distribution and the properties of these sets are more or less the same. Or stated differently, these sets will have approximately the same values for features like cardinality and label dependency.

Thus, the focus of the generator are the hypotheses for ML classification. These hypotheses are formed by a group of hyperplanes that split the input space in a positive and a negative region. In fact, a set of hyperplanes may define a linear classifier or a nonlinear one. We build nonlinear tasks in this work.

For this purpose, we assign relevant labels to regions of the input space defined by the intersection of several hyperplanes that share a common point. In other words, the relevant labels are geometrically defined at the interior of pyramids with a certain number of faces. In all cases,

$$\mathcal{X} = [0, 1]^p.$$

Then, for a given label l_j , we define a hypothesis h_j as follows:

$$h_j(\mathbf{x}) = 1 \iff \langle \mathbf{w}_j^k, \mathbf{x} - \mathbf{x}_j^0 \rangle \geq 0, \quad \forall k = 1, \dots, \text{faces}.$$

where

$$\mathbf{x}_j^0 \in \mathcal{X}, \mathbf{w}_j^k \in [-1, 1]^p, \quad k = 1, \dots, \text{faces}.$$

However, if the list of vectors \mathbf{w}_j^k is completely random, the interior of the pyramid may be empty or too small. To avoid these possibilities we force the list of \mathbf{w}_j^k to form angles within a given range, using the following procedure. First, a set of vectors ($\mathbf{w}_j^k : k = 1, \dots, \text{faces}$) is randomly drawn in $[-1, 1]^p$. Then, using the Gram–Schmidt procedure, we obtain an orthonormal basis of the linear span of vectors \mathbf{w}_j^k . Let

$$(\mathbf{v}_j^k : k = 1, \dots, \text{faces}) \leftarrow$$

$$\text{Gram-Schmidt}(\mathbf{w}_j^k : k = 1, \dots, \text{faces})$$

If the rank of vectors \mathbf{w}_j^k is not equal to faces, a new set is drawn. The next step redefines the vectors of the nonlinear hypothesis as follows:

$$\mathbf{w}_j^1 \leftarrow \mathbf{v}_j^1$$

$$\mathbf{w}_j^k \leftarrow \lambda \mathbf{v}_j^1 + \mathbf{v}_j^k, \quad k = 2, \dots, \text{faces}$$

Notice that

$$\cos(\mathbf{w}_j^1, \mathbf{w}_j^k) = \frac{\langle \mathbf{v}_j^1, \lambda \mathbf{v}_j^1 + \mathbf{v}_j^k \rangle}{\sqrt{1 + \lambda^2}} = \frac{\lambda}{\sqrt{1 + \lambda^2}}$$

Therefore,

$$\frac{\cos^2(\mathbf{w}_j^1, \mathbf{w}_j^k)}{1 - \cos^2(\mathbf{w}_j^1, \mathbf{w}_j^k)} = \frac{1}{\tan^2(\mathbf{w}_j^1, \mathbf{w}_j^k)} = \lambda^2$$

and hence it is straightforward to fix a range for λ values if we want that, for instance,

$$\text{angle}(\mathbf{w}_j^1, \mathbf{w}_j^k) \in [50^\circ, 80^\circ], \quad \forall k = 2, \dots, \text{faces}.$$

Notice that the interior angle of pyramid faces with the first one will range in $[100^\circ, 130^\circ]$. In the experiments reported at the end of the paper, the number of faces will be set to 5.

6.2 Conditional dependency

To obtain a dataset with a certain degree of *conditional dependency*, we can use the following method of two steps. If L is the number of labels required, first we use the procedure described above to search for a dataset with $L/2$ labels. Let

$$D_1 = (\mathbf{X}, \mathbf{Y}_{L/2})$$

be such a dataset. Thus, to obtain the rest of the labels, we use the whole dataset D_1 as the input instances and search for a new collection of $L/2$ labels. In this way, we have

$$D_2 = ((\mathbf{X}, \mathbf{Y}_{L/2}), \mathbf{Y}'_{L/2}).$$

At the end, we obtain

$$D = (\mathbf{X}, [\mathbf{Y}_{L/2} \ \mathbf{Y}'_{L/2}]),$$

a dataset with L labels and some degree of conditional dependency.

To ensure a cardinality similar to a given amount set by the user, we divide the cardinality into two equal parts: one part for the matrix $\mathbf{Y}_{L/2}$ searching for D_1 , and the rest for the second half for the matrix, $\mathbf{Y}'_{L/2}$, searching for D_2 .

On the other hand, the unconditional dependency cannot be guaranteed. Thus, we ask for the same amount in both searches to reach a similar value at the end of the process.

7 Experiments

Several experimental studies in the literature based on benchmark datasets, see for instance [17] and [11], report a better performance of new ML methods with respect to BR in terms of some loss functions, including macro-average measures. Most of these new approaches are aimed at exploiting label dependency. The conclusion of these studies is that the improvement is due to overcoming the main drawback of BR, the label independence assumption.

However, as we previously discussed in Sect. 5, benchmark datasets are somehow limited in several aspects. The main idea of our experiments is to make a broader comparison between BR and a recent ML method. The aim is to prove if the better performance of this new ML learner on benchmark datasets still remains in other domains, in which

we can control some important properties for ML classification, such as the number of labels, the cardinality and, more importantly, the label dependency.

We compared the scores achieved by BR with those obtained by ECC (*Ensembles of Classifier Chains*). This is a recent ML learner based on *Classifier Chains* (CC) [17] that performs particularly well in several studies. CC, designed to take advantage of label dependencies, learns L binary classifiers linked along a chain, where each classifier deals with the binary relevance problem associated with one label. In the training phase, the feature space of each classifier is extended with the actual label information of all previous labels in the chain. For instance, if the chain follows the order $l_1 \rightarrow l_2 \rightarrow \dots \rightarrow l_L$, then the classifier h_j responsible for predicting the relevance of l_j is of the form

$$h_j : \mathcal{X} \times \{0, 1\}^{j-1} \longrightarrow \{0, 1\},$$

and the training data for this classifier consists of instances $(\mathbf{x}_i, y_{i,1}, \dots, y_{i,j-1})$ labeled with $y_{i,j}$, that is, original training instances \mathbf{x}_i supplemented by the relevance of the labels l_1, \dots, l_{j-1} preceding l_j in the chain.

At prediction time, when a new instance \mathbf{x} needs to be labeled, label predictions are produced by successively querying each classifier h_j . Note, however, that the inputs of these classifiers are not well defined, since the supplementary attributes $(y_{i,1}, \dots, y_{i,j-1})$ are not available. These missing values are therefore replaced by their respective predictions made by previous classifiers along the chain.

The main drawback of CC is that it depends on the ordering of the labels in the chain. This problem can be solved using an ECC because each CC model in the ensemble uses a different label order. The final posterior probability for a label is given by the average of the posterior probabilities produced by each CC model for that label.

7.1 Experimental setting

We employed a total of 84 synthetic datasets for this experiment; each dataset was formed by a training, a validation and a test set. First, we generated 42 noise-free datasets with conditional dependency using the generator described in Sect. 6. Then, another 42 datasets were obtained from them by adding artificial noise using a Bernoulli distribution, that is, the labels of both training and validation sets swapped their values with a probability 0.01. The properties of the datasets generated are: the cardinality ranges in [4..9], the number of labels belongs to {10, 25, 50, 75, 100, 150, 200}, and the label dependence is in [0.1..0.35], measured using (11).

The base learner for both methods was SVM [2] with a Gaussian kernel (RBF). The parameters C and γ (for the Gaussian kernel) were adjusted with a grid search using the validation dataset generated. The parameters

Table 3 Average F_1^{mi} scores in test sets for datasets generated explicitly with conditional dependency, see Sect. 6.2

#Labels	BR (%)	ECC* (%)
Noise free		
10	83.88	84.46
25	81.12	81.76
50	76.34	77.46
75	72.29	72.38
100	69.43	69.68
150	63.31	62.77
200	58.43	56.87
Noise added		
10	83.60	84.11
25	80.59	81.08
50	75.06	76.68
75	70.68	71.05
100	67.00	67.37
150	61.55	60.20
200	54.53	54.96

could vary in $C \in \{10^i : i = -1, \dots, 3\}, \gamma \in \{10^{-3}, 10^{-2}, 10^{-1}, 0.3, 0.5, 1\}$. For ECC, we used the implementation by [4], denoted as ECC*. This means that, unlike [17], we did not apply any threshold selection method for deciding the relevance of a label. Of course, the same policy was applied for BR. The number of CC models in a ECC* classifier was set to 10.

To compare the performance of BR and ECC* on this collection, we used the micro-average F_1 scores (4). We selected F_1^{mi} for several reasons. On the one hand, we did not employ Hamming loss or macro-average measures because BR optimizes such measures when a proper base learner is used. On the other hand, ECC* does not optimize any particular measure, mainly because it is an ensemble method. Thus, we selected an intermediate measure in which ECC* seems to outperform BR, according to previous studies cited at the beginning of this section, with basically the same experimental setup. Also, ECC* is among the best methods in terms of F_1^{mi} in the results reported by [10], performing better than BR.

7.2 Experimental results

Table 3 shows the average scores achieved by BR and ECC*. Additionally, Table 4 summarizes the significant differences between both learners using a Wilcoxon two-sided signed rank test.

The first evidence is that ECC* is significantly better than BR with a low number of labels, both for noise-free and noisy datasets. These results are quite coherent with those reported by [17]. In that paper, the experimental results were

Fig. 1 Dataset cardinality. In the X-axis is the cardinality, while in Y-axis the difference in terms of F_1^{mi} between ECC* and BR. Each point represents the results for a dataset and when it is above 0 indicates that ECC* outperforms BR

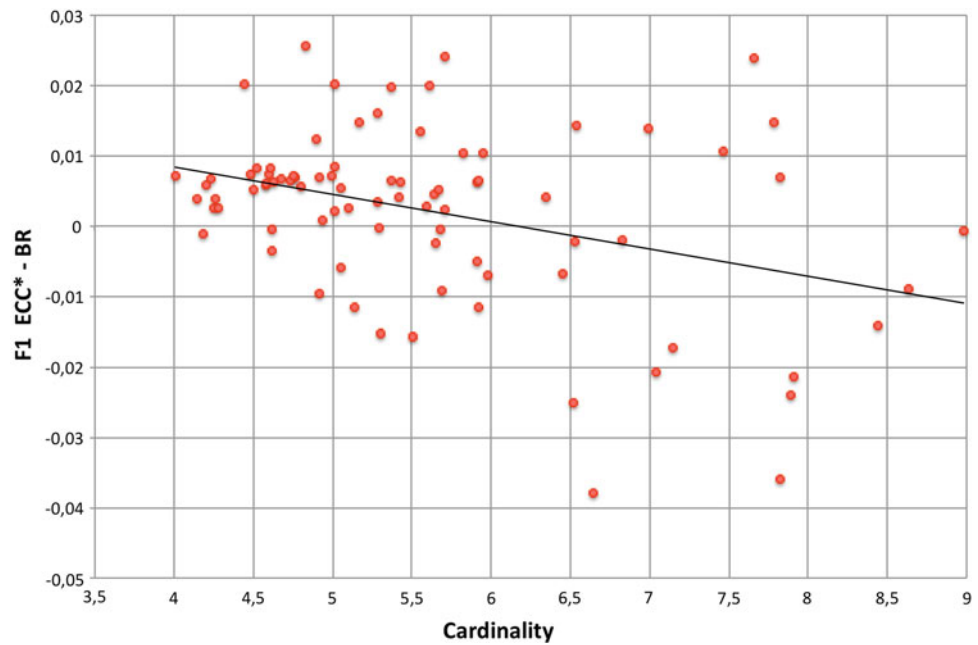


Table 4 Significant differences for several numbers of labels using a Wilcoxon two-sided signed rank test

Noise	Range of labels	Significant
Free	{10, 25, 50}	$ECC^* \gg BR$
	{75, 100}	$ECC^* \cong BR$
	{150, 200}	$ECC^* \ll BR$
Added	{10, 25, 50, 75}	$ECC^* \gg BR$
	{100, 150, 200}	$ECC^* \cong BR$

The p values in significant differences (\gg or \ll) were always below 0.01. For those cases in which the difference was not significant (\cong), the obtained p value was greater than 0.60

made with 15 datasets; only two of them have more than 103 labels.

Nevertheless, the differences between the two methods become smaller when the number of labels increases; even BR significantly outperforms ECC* for noise-free datasets and more than 100 labels. Maybe the reason is the accumulation of errors forced by the Classifier Chains algorithm of ECC*, which is more likely to happen when the number of labels is large. This result could not be obtained using only benchmark datasets, simply because there are not enough datasets to statistically support this conclusion.

Our experiments, based on synthetic datasets, allow us to analyze more aspects. For instance, Fig. 1 depicts the performance of both methods with respect to the cardinality. Each point shows the results for a dataset in which X-axis represents the cardinality and Y-axis the difference in terms of F_1^{mi} between ECC* and BR. Thus, a point above 0 in the

Y-axis indicates that ECC* outperforms BR in that dataset. The graphic demonstrates that for those datasets with lower cardinality ECC* is much better, but when the cardinality is higher the result is just the opposite, BR ameliorates the scores of ECC*.

But the most interesting analysis is perhaps that which studies the performance in function of label dependency; see Fig. 2. This graphic is equivalent to the previous one: Y-axis represents again the difference in terms of F_1^{mi} between ECC* and BR, but now the X-axis stands for the label dependency of the datasets measured applying (11). Despite the results seem quite mixed, the tendency line shows again that the scores of ECC* tend to be worse in comparison to those of BR when the label dependency increases. With a low label dependency, ECC* is clearly better, but for larger values BR is able to be at least competitive and sometimes superior.

Finally, the scores of each of the 84 datasets are also reported graphically in Fig. 3. The goal is to represent somehow the complexity of the datasets, measured in terms of F_1^{mi} : the greater the F_1^{mi} value, the easier is the learning task. In this case, the meaning of the axes is different. Each point represents the results for a dataset, but now X-axis is the F_1^{mi} score for BR, and Y-axis the F_1^{mi} result for ECC*. Therefore, those points above the diagonal correspond to datasets in which ECC* outperforms BR, and the other way around. When the tasks are easier, with higher F_1^{mi} results, ECC* is better, but when the F_1^{mi} scores decrease, then BR usually achieves the best results.

Actually, all these analyses reflect the same conclusion: when the learning task is easier (less cardinality or less label dependency or a fewer number of labels), ECC* performs

Fig. 2 Dataset label dependency. In the X-axis is the label dependence, while in Y-axis is the difference in terms of F_1^{mi} between ECC* and BR. Each point represents the results for a dataset and when its above 0 indicates that ECC* outperforms BR

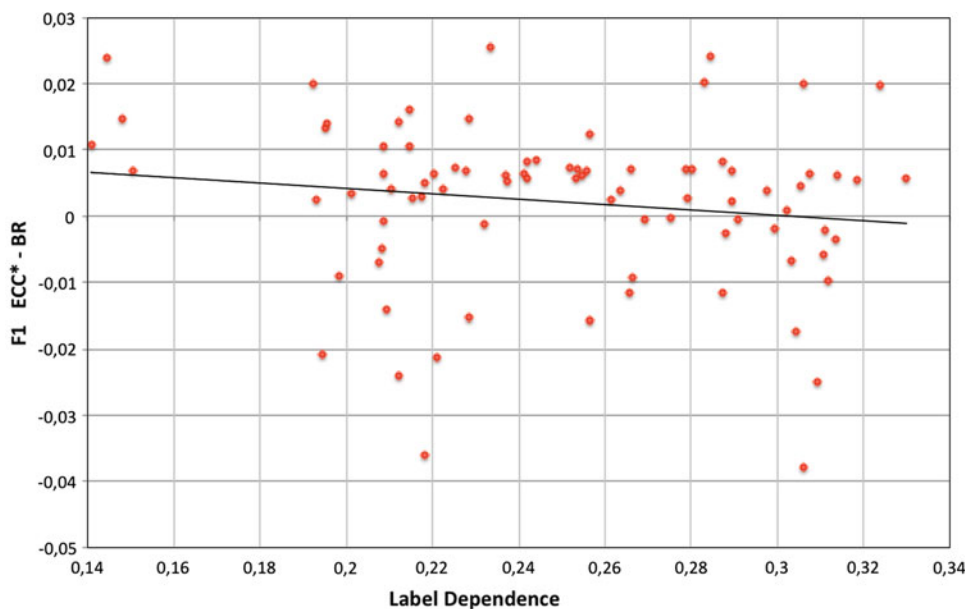
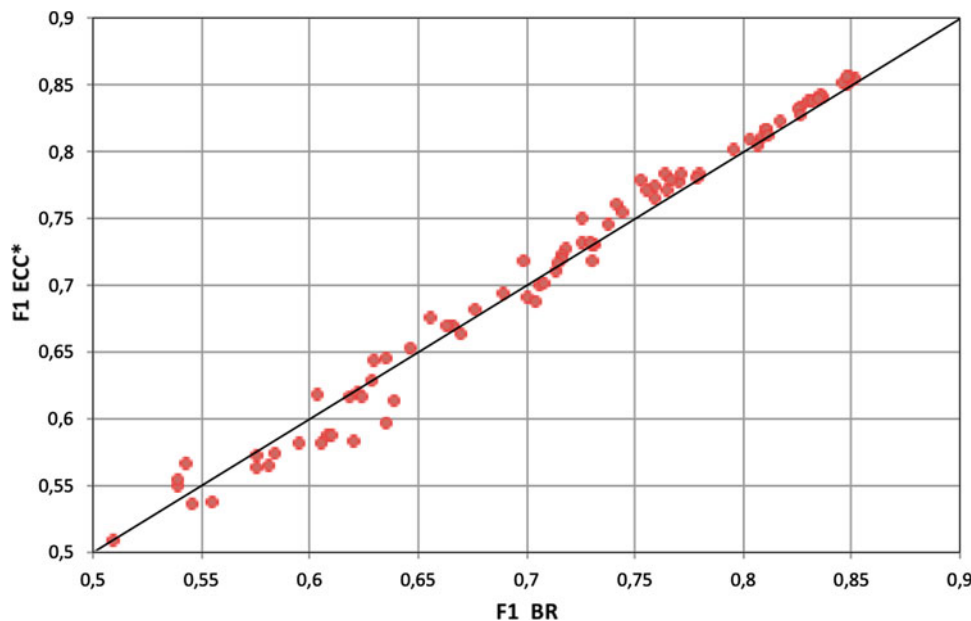


Fig. 3 Dataset Complexity. Each point is a pair of F_1^{mi} scores achieved in the same dataset by BR and ECC*. Points above the diagonal represent datasets where ECC* outperforms BR



better. But when the domain is more complex, with more labels or a greater cardinality or label dependency, then BR is at least competitive and sometimes superior.

The purpose of these experiments is not just to analyze BR and ECC* from another perspective. The goal was also to point out that it is difficult to extract useful conclusions, statistically supported, using only benchmark datasets; there are too few domains given the complexity of ML classification. This being the case, synthetic datasets may allow us to gain more insight into the behavior of ML methods and analyze them with respect to different factors, as shown in this study.

8 Conclusions

In this article, we tried to demystify some clichés about ML classification and its main baseline method, binary relevance. For instance, one interesting point is to acknowledge the properties of BR, not only its computational complexity, but also that it is well tailored to produce good ML classifiers for several ML loss functions. Hamming loss and macro-averages are clearly oriented to the use of learners that consider each label separately. A correct implementation of BR, using the appropriate base learner for the target loss function, should be enough if one wants to achieve good scores.

Thus the main efforts of ML community should be focused on devised methods for optimizing other kind of measures.

New proposals can obviously improve the performance of BR for other performance metrics. However, the experimental studies are limited due to the lack of benchmark domains. There are just a few publicly available domains and they cover a small and biased proportion of the huge possibilities of ML datasets. Under these circumstances, our proposal is to combine benchmark with synthetic datasets to perform more complete experimental studies. In this paper, we have used an ML dataset generator that produces synthetic domains in which the user can select properties like the number of labels, the cardinality and the label dependency.

We have compared the efficacy of BR and an Ensemble of Classifier Chains using a collection of synthetic problems. The main conclusion is that the performance of ECC* dramatically drops when the complexity of the dataset increases—a larger number of labels or a greater cardinality or a higher label dependency—while BR is quite competitive under these circumstances. This is only an example of a situation that current experimental settings based on the benchmark datasets available are not able to detect.

Acknowledgments The research reported here is supported in part under grant TIN2011-23558 from the Ministerio de Economía y Competitividad, Spain. We would also like to acknowledge all those people who generously shared the datasets and software used in this paper.

References

- Bielza, C., Li, G., Larrañaga, P.: Multi-dimensional classification with bayesian networks. *Int. J. Approx. Reason.* **52**(6), 705–727 (2011)
- Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2**, 27:1–27:27. Available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/> (2011)
- Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. *Mach. Learn.* **76**(2), 211–225 (2009)
- Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: Proceedings of the 27th International Conference on Machine Learning (ICML) pp. 279–286 (2010)
- Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: An exact algorithm for f-measure maximization. In: Proceedings of the Neural Information Processing Systems (NIPS), pp. 1404–1412 (2011)
- Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: *Advances in Neural Information Processing Systems 14*, MIT Press, Cambridge, pp. 681–687 (2001)
- Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, vol. 3056, pp. 22–30 (2004)
- Joachims, T.: A support vector method for multivariate performance measures. In: *Proceedings of the ICML '05*, pp. 377–384 (2005)
- Lastra, G., Luaces, O., Quevedo, J., Bahamonde, A.: Graphical feature selection for multilabel classification tasks. In: Gama, J., Bradley, E., Hollmén, J. (eds.) *Proceedings of Advances in Intelligent Data Analysis X (IDA 2011)*. Springer, Lecture Notes in Computer Science, vol. 7014, 246–257 (2011)
- Madjarov, G., Kocev, D., Gjorgjević, D., Deroski, S.: An extensive experimental comparison of methods for multi-label learning. *Pattern Recognit.* **45**(9), 3084–3104 (2012)
- Montañés, E., Quevedo, J., del Coz, J.: Aggregating independent and dependent models to learn multi-label classifiers. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECLM-PKDD)* pp. 484–500 (2011)
- Petterson, J., Caetano, T.: Reverse multi-label learning. *Adv. Neural Inform. Process. Syst.* **23**, 1912–1920 (2010)
- Quevedo, J.R., Luaces, O., Bahamonde, A.: Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognit.* **45**(2), 876–883 (2012)
- Read, J., Pfahringer, B., Holmes, G.: Generating synthetic multi-label data streams. In: *ECML/PKDD 2009 Workshop on Learning from Multi-label Data (MLD'09)*, pp. 69–84 (2009a)
- Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pp. 254–269 (2009b)
- Read, J., Bifet, A., Holmes, G., Pfahringer, B.: Streaming multi-label classification. *JMLR Workshop and Conference Proceedings (Second Workshop on Applications of Pattern Analysis)*, vol. 17, pp. 19–25 (2011a)
- Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. *Mach. Learn.* **85**(3), 333–359 (2011b)
- Schapire, R., Singer, Y.: Boostexter: a boosting-based system for text categorization. *Mach. Learn.* **39**(2), 135–168 (2000)
- Tsoumakas, G., Katakis, I.: Multi label classification: an overview. *Int. J. Data Warehousing Min.* **3**(3), 1–13 (2007)
- Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multilabel data. *Data Mining and Knowledge Discovery Handbook* pp. 667–685 (2010a)
- Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multi-label classification. *IEEE Trans. Knowl. Discov. Data Eng.* **23**(7), 1079–1089 (2010b)
- Zaragoza, J., Sucar, L., Bielza, C., Larrañaga, P.: Bayesian chain classifiers for multidimensional classification. In: *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2192–2197 (2011)
- Zhang, M.L., Zhou, Z.: ML-KNN: a lazy learning approach to multi-label learning. *Pattern Recognit.* **40**(7), 2038–2048 (2007)