Pattern Recognition 47 (2014) 1494–1508

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Dependent binary relevance models for multi-label classification

Elena Montañes [a], Robin Senge [b], Jose Barranquero [a], José Ramón Quevedo [a], Juan José del Coz [a,*], Eyke Hüllermeier [b]

[a] Artificial Intelligence Center, University of Oviedo at Gijón, 33204 Gijón, Spain
[b] Mathematics and Computer Science, Marburg University, 35032 Marburg, Germany

ARTICLE INFO

Article history:
Received 31 May 2013
Received in revised form
24 September 2013
Accepted 25 September 2013
Available online 4 October 2013

Keywords:
Multi-label classification
Label dependence
Stacking
Chaining

ABSTRACT

Several meta-learning techniques for multi-label classification (MLC), such as chaining and stacking, have already been proposed in the literature, mostly aimed at improving predictive accuracy through the exploitation of label dependencies. In this paper, we propose another technique of that kind, called *dependent binary relevance* (DBR) learning. DBR combines properties of both, chaining and stacking. We provide a careful analysis of the relationship between these and other techniques, specifically focusing on the underlying dependency structure and the type of training data used for model construction. Moreover, we offer an extensive empirical evaluation, in which we compare different techniques on MLC benchmark data. Our experiments provide evidence for the good performance of DBR in terms of several evaluation measures that are commonly used in MLC.

1. Introduction

Multi-label classification (MLC) is a machine learning problem in which models are sought that assign a subset of (class) labels to each object, unlike conventional (single-class) classification that involves predicting only a single class. Multi-label classification problems are ubiquitous and naturally occur, for instance, in assigning keywords to a paper, tags to resources in a social network, objects to images or emotional expressions to human faces.

There is a considerable amount of literature, in which state-of-the-art binary or multi-class classification algorithms are adapted and extended to the setting of MLC, including methods using decision trees [1], instance-based algorithms [2], neural networks [3], support vector machines [4], naive Bayes [5], conditional random fields [6] and boosting [7]. Besides, there is also another line of research, in which approaches of that kind are completely put aside; instead, the development of specialized methods that consider the particularities of multi-label data is advocated.

In general, the problem of multi-label learning is coming with two fundamental challenges. The first one bears on the computational complexity of the algorithms. If the number of labels is large, then a complex approach might not be applicable in practice. Therefore, the scalability of algorithms is a key issue in this field. The second problem is related to the very nature of multi-label data. Not only is the number of classes typically larger than in multi-class classification tasks, but also each example belongs to a variable-sized subset of labels simultaneously. Moreover, and perhaps even more importantly, the labels will normally not occur independent of each other; instead, there are statistical dependencies between them. From a learning and prediction point of view, these relationships constitute a promising source of information, in addition to that coming from the mere description of the objects. Thus, it is hardly surprising that research on MLC has very much focused on the design of new methods that are able to detect—and benefit from—interdependencies among labels.

In recent years, many papers have analyzed the presence of label correlations, including theoretical analyses of label dependence in the context of MLC [8]. In this regard, different types of dependence have been formally distinguished, such as conditional dependence [6,9–12] and marginal (unconditional) dependence [3,13,14]. Other papers are aiming at the exploitation of relations in different sets of labels, such as pairwise relations [3,4,7,15,16], relations in sets of different sizes [11,17,18], or relations in the whole set of labels [10,13,14]. Exploiting label dependence implicates the induction of complex models. In fact, the more the label combinations are considered, the more complex the models are. This does not mean that exploiting pairwise correlations is preferable to exploiting full-order correlations, since the former may fail to capture the true dependencies while the latter may not work well if the labels display complex relations that are difficult to deal with.

This paper proposes *dependent binary relevance* (DBR) models as an efficient and effective approach to induce multi-label classifiers that exploit conditional label dependence. Instead of studying them in combination with independent classifiers, like in [10], our goal is

* Corresponding author. Tel.: +34 985182501; fax: +34 985182125.
E-mail address: juanjo@aic.uniovi.es (J. José del Coz).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.09.029

to explore their behavior when used in isolation, extending the work presented in [19] in which this approach was favorably compared with several state-of-the-art methods [3,11,13,18]. The DBR approach is conceived as a natural extension of the simple binary relevance strategy, which does not allow for exploiting conditional label dependence. We shall elaborate on the positioning of our approach more closely in Section 3, where we argue that this approach combines properties of two other meta-techniques for MLC, namely chaining [11] and stacking [14], and that it fills a "gap" within the spectrum of methods that have been devised so far.

A key contribution of this paper is a deep analysis of the properties of dependent binary models, in which we characterize those conditions under which they should work well in practice. These models require label estimations (produced by any multi-label classifier) at prediction time. This issue is analyzed throughout the paper, concluding that the more reliable these estimations are, the better the overall performance becomes.

Another contribution of this work is to present a comprehensive study of methods based on chaining [11] and stacking [14] strategies. Our goal is to analyze these two approaches, which are closely connected, and to study those factors that have an influence on their performance. A key distinction between both approaches is the type of training data they rely on, which in turn has a decisive impact on the kind of label dependence captured.

The rest of the paper is organized as follows. The next section introduces multi-label classification in a more formal way. Stacking and chaining methods are reviewed in Section 3. Section 4 is devoted to the new DBR technique; we describe this approach formally and provide a detailed analysis of its properties. Finally, experimental results are reported in Section 5, before concluding the paper in Section 6.

2. Multi-label classification

Before describing some previous approaches to tackle multi-label classification, we present this learning task in a more formal way. The point of departure is a finite and non-empty set of labels $\mathcal{L}=\{\ell_1,\ell_2,\ldots,\ell_m\}$ and a training set $S=\{(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_n,\boldsymbol{y}_n)\}$. The elements of this set are supposed to be independently and randomly drawn according to an unknown probability distribution $\mathbf{P}(\mathbf{X},\mathbf{Y})$ on $\mathcal{X}\times\mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and the output space respectively. The former is the space of the object descriptions (instances), whereas the latter is given by the power set $\mathcal{P}(\mathcal{L})$ of $\mathcal{L}$. To ease notation, we define $\boldsymbol{y}_i$ as a binary vector $\boldsymbol{y}_i=(y_{i,1},y_{i,2},\ldots,y_{i,m})$ in which $y_{i,j}=1$ indicates the presence (relevance) and $y_{i,j}=0$ the absence (irrelevance) of ℓ_j in the labeling of $\boldsymbol{x}_i$. Using this convention, the output space can also be defined as $\mathcal{Y}=\{0,1\}^m$. The goal in MLC is to induce from S a hypothesis $\boldsymbol{h}:\mathcal{X}\longrightarrow\mathcal{Y}$ that correctly predicts the subset of relevant labels for unlabeled query instances $\boldsymbol{x}$.

The most straightforward and arguably simplest approach to tackle multi-label classification is *binary relevance* (BR). The BR strategy reduces a given multi-label problem with m labels to m binary classification problems. More precisely, m hypotheses $h_1,h_2,\ldots,h_m$ are induced, each of them being responsible for predicting the relevance of one label, using just $\mathcal{X}$ as the input space:

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

In this way, the labels are predicted independent of each other and no label dependencies are taken into account. Yet, despite its inability to exploit any label dependencies, the BR algorithm also exhibits several advantages: (i) each binary learning method can be used as base learner, (ii) it has linear complexity with respect to the number of labels and (iii) it can be easily parallelized.

In spite of its simplicity, the BR method obtains competitive results in benchmark datasets whenever being applied on top of a state-of-the-art base learner with a proper procedure for tuning parameters. Interestingly, it has been shown theoretically and empirically that BR performs quite strong in terms of decomposable loss functions [9]. This behavior can be explained by studying BR from a probabilistic point of view. Given that each binary model h_j is able to estimate $\mathbf{P}(y_j|\boldsymbol{x})$, BR is well-suited for every loss function whose risk minimizer can be expressed in terms of marginal distributions of labels. Since the classifier used for learning h_j commonly optimizes its accuracy, the whole BR model minimizes the Hamming loss[1]:

$$\text{HammingLoss}(\boldsymbol{y},\boldsymbol{h}(\boldsymbol{x}))=\frac{1}{m}\sum_{i=1}^{m}[\![\,y_i\neq h_i(\boldsymbol{x})\,]\!]. \tag{2}$$

This measure averages the standard 0/1 classification error over the m labels and hence corresponds to the proportion of labels whose relevance is incorrectly predicted. Besides, if an appropriate base learner is employed, then BR is also able to optimize all other macro-average label-based metrics, such as the macro-F_1 measure [20].

On the other hand, the decomposition approach followed by BR affects its performance for those loss functions whose minimization requires an estimation of the joint distribution. Examples of these measures are micro-average metrics and Subset 0/1 loss, which looks if the predicted and relevant label subsets are equal or not:

$$\text{Subset}_{0/1}(\boldsymbol{y},\boldsymbol{h}(\boldsymbol{x}))=[\![\,\boldsymbol{y}\neq\boldsymbol{h}(\boldsymbol{x})\,]\!]. \tag{3}$$

In these cases, it is necessary to develop algorithms which are able to estimate the joint label probability distributions to obtain predictions that minimize this sort of metrics. Dembczyński et al. [8,21] present a formal probabilistic analysis of multi-label classification, studying the connection between risk minimization and loss functions.

3. Modeling label dependence

The arguably most natural way to capture label dependencies is to learn classifier models that *condition* the prediction of a label y_i not only on the object features $\boldsymbol{x}$ but also on some of the other labels y_j. This idea of conditioning can be realized in different ways. In particular, the following distinctions can be made:

(i) *Full vs. partial conditioning*: The prediction of y_i can be conditioned on all other labels $\{y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_m\}$ or only on a subset of these labels. The most "sparse" conditioning scheme among those that capture full dependence between labels is a sequential structure: y_i is conditioned on $\{y_1,\ldots,y_{i-1}\}$. This structure, which constitutes the core of the idea of *classifier chains* (to be discussed further below), can be motivated by the product rule of probability [9]:

$$\mathbf{P}(\boldsymbol{y}|\boldsymbol{x})=\prod_{i=1}^{m}\mathbf{P}(y_i|\boldsymbol{x},y_1,\ldots,y_{i-1}) \tag{4}$$

(ii) *True vs. predicted label information*: For training the predictor of y_i, the other labels y_j are available in the training data and, therefore, can in principle be used for learning this predictor. Alternatively, the model for y_i can be trained on the estimations $\hat{y}_j$ produced by the other predictors.

[1] The expression $[\![\,p\,]\!]$ evaluates to 1 if the predicate p is true, and to 0 otherwise.

As for (i), a dependency structure in the form of a chain, i.e., conditioning y_i on $\{y_1,\ldots,y_{i-1}\}$, is quite appealing. In fact, as already mentioned, this structure is sufficient to capture full label dependence. Moreover, among these structures, it is the only one without any cycles in the underlying dependency graph. Thus, it is also theoretically sound and unproblematic from an inference point of view. Therefore, one may wonder whether there is any reason to go beyond a sequential dependency structure.

One reason for an affirmative answer to this question is of practical nature: Conditional probabilities, like those in (4), need to be estimated by corresponding models induced from the training data, and the quality of these models may strongly depend on which kind of mapping they are supposed to implement. Moreover, by using a dependency structure that is more complete than a chain (and therefore in a sense "over-complete"), a certain level of redundancy is introduced, which might again be beneficial from a practical point of view (see Fig. 1).

As for (ii), one may likewise wonder whether there is any reason to ignore the true label information y_j at training time, using predictions $\hat{y}_j$ instead. Again, however, there is a motivation for doing so: Since true label information y_j is only available for training, this information has to be replaced by estimations $\hat{y}_j$ at prediction time. Therefore, since the distribution of the true labels will normally differ from the distribution of the predicted ones, the original training data is not representative of the test data. Consequently, a classifier learned on the observed labels y_j might be misled by the training data [22]. This problem is avoided by training classifiers on the predictions $\hat{y}_j$ right away. This approach, however, makes only partial use of the training data and, moreover, may fail to discover the true label dependencies.

From our discussion so far, we conclude that none of the four combinations resulting from the above distinctions, namely

(a) partial conditioning, true label information,
(b) partial conditioning, predicted label information,
(c) full conditioning, true label information,
(d) full conditioning, predicted label information,

should be dismissed immediately. The topic of this paper is the only variant that is still missing in this list, namely (c). Before turning to this variant, we briefly recall the methods that have been proposed for the other alternatives.

3.1. Classifier chains

The *Classifier Chains* (CC) method introduced by Read et al. [11] implements alternative (a). As its name suggests, CC selects an order on the label set—a *chain* of labels—and trains a binary classifier for each label following this order. In the training phase, the feature space of each classifier in the chain is extended with the true label information of all previous labels in the chain. For instance, if the chain follows the order $\ell_1 \to \ell_2 \to \cdots \to \ell_m$, then the classifier h_j responsible for predicting the relevance of ℓ_j is of the form:

$$h_j : \mathcal{X} \times \{0,1\}^{j-1} \longrightarrow \{0,1\}. \tag{5}$$

The training data for this classifier consists of instances $(\boldsymbol{x}_i, y_{i,1}, \ldots, y_{ij-1})$ labeled with y_{ij}, that is, original training instances $\boldsymbol{x}_i$ supplemented by the relevance of the labels $\ell_1,\ldots,\ell_{j-1}$ preceding ℓ_j in the chain.

At prediction time, when a new instance $\boldsymbol{x}$ needs to be labeled, a prediction $\hat{\boldsymbol{y}} = (\hat{y}_1,\ldots,\hat{y}_m)$ is produced by successively querying each classifier h_j. As already mentioned, however, the inputs of these classifiers are not well-defined, since the true attributes $y_1,\ldots,y_{j-1}$ are not available at prediction time. These missing values are therefore replaced by their respective predictions: y_1 used by h_2 as an additional input is replaced by $\hat{y}_1 = h_1(\boldsymbol{x})$, y_1 and y_2 used by h_3 as an additional inputs are respectively replaced by $\hat{y}_1 = h_1(\boldsymbol{x})$ and $\hat{y}_2 = h_2(\boldsymbol{x}, \hat{y}_1)$, and so forth. Thus, the prediction for an object $\boldsymbol{x}$ is of the form:

$$\boldsymbol{y} = (h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, h_1(\boldsymbol{x})), h_3(\boldsymbol{x}, h_1(\boldsymbol{x}), h_2(\boldsymbol{x}, h_1(\boldsymbol{x}))), \ldots).$$

As we also mentioned before, the models h_j are not perfect predictors. Thus, although different chains (label orders) should theoretically produce the same results,[2] their performance will normally differ in practice. Read et al. [11] therefore propose the use of an *ensemble of classifier chains* (ECC), in which different chains (random permutations of the labels) are trained and their predictions combined. The observation that ECC typically outperforms CC can be seen as an example of the redundancy effect mentioned above: Although a single chain should theoretically suffice, there is a practical benefit from exploiting and combining more conditional dependencies.

3.1.1. Probabilistic classifier chains

A probabilistic variant of classifier chains was proposed by Dembczyński et al. [9]. Probabilistic Classifier Chains (PCC) are inspired by the product rule (4) and implement this rule by training corresponding probabilistic classifiers. Thus, unlike CC that predicts only a single label combination, PCC estimates the entire joint distribution $\mathbf{P}(\cdot|\boldsymbol{x})$. By doing so, PCC generally improves in terms of accuracy, albeit at the cost of a higher computational complexity. Recently, two approaches have been proposed to reduce this complexity. In [23], the authors show that a probabilistic variant of chaining becomes tractable using Monte Carlo sampling. The method presented in [24] applies beam search, both to perform tractable test time inference with PCC and to determine a suitable label order.

3.2. Stacking

The use of the stacked generalization learning paradigm [25], or simply stacking, in the context of multi-label classification is a way to implement alternative (d). It was first proposed by Godbole and Sharawagi [14]. In the learning phase, their method builds a stack of two groups of classifiers. The first one is formed by the BR classifiers:

$$\boldsymbol{h}^1(\boldsymbol{x}) = (h_1^1(\boldsymbol{x}), \ldots, h_m^1(\boldsymbol{x})).$$

In a second level, also called meta-level, another group of binary models is learned, again one for each label. These classifiers consider an augmented feature space that includes the binary outputs of all models of the first level:

$$\boldsymbol{h}^2(\boldsymbol{x}, \hat{\boldsymbol{y}}) = (h_1^2(\boldsymbol{x}, \hat{\boldsymbol{y}}), \ldots, h_m^2(\boldsymbol{x}, \hat{\boldsymbol{y}})),$$

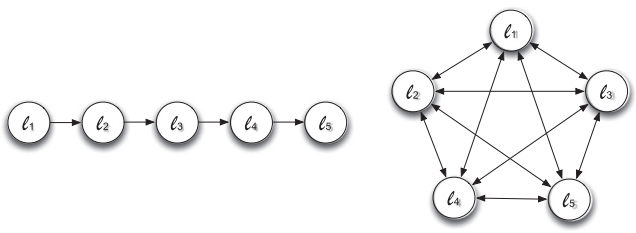

Fig. 1. Full vs. partial conditioning for modeling label dependencies. The sequential structure on the left is the most "sparse" conditioning scheme among those that capture dependence between all labels. The full conditioning scheme on the right is redundant and contains cycles.

[2] Indeed, note that the product rule (4) holds true regardless of the order.

where $\hat{\boldsymbol{y}} = \boldsymbol{h}^1(\boldsymbol{x})$. Obviously, the meta-level classifiers are supposed to induce the dependencies between the labels. At prediction time, the MLC model returns the outputs of the meta-level classifiers, $\boldsymbol{h}^2(\boldsymbol{x})$, while the predictions of $\boldsymbol{h}^1(\boldsymbol{x})$ are only used to obtain the attribute values in the extended feature space.

Some variants of the stacking approach have been proposed, mainly aimed at reducing the augmented feature space by removing some label dimensions. The idea is to ignore the information of those labels that are not related with label ℓ_j in order to induce the meta-model $h_j{}^2$. For instance, in [26] the authors propose to calculate the chi-coefficient between each pair of labels, (ℓ_j, ℓ_k), based on an initial single pass over the training set. The method prunes the information of labels ℓ_k with a correlation below a threshold. The idea of modeling label dependence in a sparse way is also related to the next approach, which is a kind of stacking approach that relies on a sequential dependency structure.

3.3. Nested stacking

By combining the ideas of chaining and stacking, Nested Stacking (NS) as proposed by Senge et al. [27] implements alternative (b). Roughly speaking, NS is equivalent to CC, except for using predicted labels $\hat{y}_j$ instead of the observed ones y_j in the training phase. Thus, the training data of the classifier (5) now consists of instances of the form $(\boldsymbol{x}_i, \hat{y}_{i,1}, \ldots, \hat{y}_{i,j-1})$ instead of $(\boldsymbol{x}_i, y_{i,1}, \ldots, y_{i,j-1})$.

As explained before, the motivation for training on predicted instead of observed labels is the observation that CC violates a key assumption of machine learning, namely that the training data is representative of the test data in the sense of being identically distributed. This assumption does not hold for the chained classifiers in CC, since y_j and $\hat{y}_j$ will normally not follow the same distribution. From the point of view of the classifier h_j, which uses the labels $y_1, \ldots, y_{j-1}$ as additional attributes, this problem can be seen as a problem of *attribute noise*. More specifically, we are facing the "clean training data vs. noisy test data" case, which is one of the four possible noise scenarios that have been studied quite extensively in [28]. For CC, this problem appears to be vital: Could it be that the additional label information, which is exactly what CC seeks to exploit in order to gain in performance (compared to BR), eventually turn out to be a source of impairment? Indeed, Senge et al. [27] show that nested stacking is at least competitive to CC, and even appears to be superior for specific loss functions.

4. Dependent binary relevance classifiers

Our proposal of *dependent binary relevance* (DBR) models relies on two main hypotheses: First, taking conditional label dependencies into account is important for performing well in multi-label classification tasks. Second, modeling and learning such dependencies in a redundant way, for example using cyclic graph structures instead of simple chains, may further improve performance in practice.

The first assumption is at the core of research in multi-label classification and quite uncontested in this field. In fact, the importance of capturing label dependencies has not only been confirmed in many empirical studies, but it has also been shown theoretically that simple binary relevance learning is not able to reach optimal performance for specific types of MLC loss function, even if the individual binary models are perfect [8].

The second assumption is arguably more disputable, especially as it is difficult to justify theoretically. On the other side, the practical usefulness of learning in a redundant way has been shown in many branches of machine learning, such as ensemble classification [29]. Besides, as already mentioned in Section 3.1, similar observations have already been made in multi-label classification, too, namely in connection with classifier chains: Although a single chain should in principle be enough to capture dependencies between all labels, ensembles of such chains are consistently better in practice.

4.1. Training phase

Formally, the DBR method works as follows. First, we train a model that is composed of as many binary classifiers as labels:

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = (h_1(\boldsymbol{x}, y_2, \ldots, y_m), \ldots, h_m(\boldsymbol{x}, y_1, \ldots, y_{m-1})), \tag{6}$$

where each individual classifier h_j is of the form:

$$h_j : \ \mathcal{X} \times \{0,1\}^{m-1} \longrightarrow \{0,1\} \tag{7}$$

and is induced from the training data:

$$S_j = \{((\boldsymbol{x}_i, y_{i,1}, \ldots, y_{i,j-1}, y_{i,j+1}, \ldots, y_{i,m}), y_{i,j}) | i = 1, \ldots, n\}. \tag{8}$$

Thus, the actual information of all labels except ℓ_j is used as additional features. Regarding label dependence, binary classifiers h_j are aimed at detecting full conditional label dependence, since the description of the object is always considered together with label information. From a probabilistic point of view, DBR classifiers try to estimate

$$\mathbf{P}(y_j | \boldsymbol{x}, y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m). \tag{9}$$

One may suspect that estimating this probability is more difficult than estimating $\mathbf{P}(y_j|\boldsymbol{x})$, as BR does, mainly because the feature space is larger. First, however, the dimension of the input space is anyway larger than the space of labels most of the time. Second, labels typically take binary values whereas attributes are often continuous. Third, the label space is typically quite sparse. Finally, the relation between the object descriptions and the labels is expected to be different from the relations that may exist among labels, and the hope is that the latter are less difficult to induce from data than the former.

4.2. Prediction phase

Like in the case of classifier chains, the actually observed labels y_i are available as additional features to induce the binary classifiers h_j only during the training stage, whereas this information is not given for a new query instance to be classified. Consequently, in order to make the DBR model applicable, the help of other (multi-label) classifiers is needed to produce estimates $\hat{y}_i$, which can then be used in place of the y_i.

Naturally, the simplest solution to obtain such estimations is to use BR as a base learner. In that case, DBR is somehow similar to the stacking approach. However, any other multi-label classifier can in principle be used, regardless of whether it assumes independence between labels or not. Formally, the inputs for the classifier h_j will be of the form:

$$(\boldsymbol{x}, \hat{y}_1, \ldots, \hat{y}_{i-1}, \hat{y}_{i+1}, \ldots, \hat{y}_m),$$

where the $\hat{y}_i$ are estimations provided by the selected base learner. Needless to say, the best alternative will be to choose that method which provides the most reliable label estimations, since the better the estimations $\hat{y}_i$ mimic the true labels y_i on which DBR is trained, the better the final predictions of DBR will be.

Applying this testing procedure, DBR is clearly an extension of the BR strategy using more information. Thus, we should expect that DBR obtains better marginal estimations for each label whenever some conditional label dependence is present. If this is true, DBR should perform well in terms of those measures whose risk minimizer can be expressed solely in terms of marginal

distributions, for instance Hamming loss or macro-average measures if an appropriate base learner is employed. The difference with respect to BR is that DBR learns from a different feature space. Remember, however, that the level of noise of the label estimations $\hat{y}_i$ used at prediction time may indeed prevent DBR from minimizing such loss functions.

Despite its simplicity, DBR appears to be a promising approach, especially as it exploits all the information available. It does not include any additional procedure, like a feature selection method for the label space or a preprocessing step that seeks to capture the label dependencies, for instance using a directed acyclic graph. A priori, this could be seen as a drawback, but in fact, it makes the method flexible enough for letting the learning process induce the necessary relations, especially given the sparseness of the representation in the label space. Furthermore, in this way the DBR approach can be studied more easily as a natural extension of BR strategy with more information.

4.3. Comparison with related approaches

Throughout this section, we will suppose that DBR is built using BR to obtain label estimations $\hat{y}_j$. This makes it more similar to related approaches such as stacking and, moreover, simplifies our analysis. In order to better understand the properties of DBR, it is useful to analyze the main differences with respect to the most related methods described above, namely Classifier Chains (CC), Stacking (STA) and Nested Stacking (NS). As already explained in Section 3, these methods can be distinguished along two main dimensions: (i) the training data they use (true vs. estimated labels) and (ii) the size of the subsets of labels for which dependencies are modeled (partial/sequential vs. full conditioning). Fig. 2 summarizes this categorization.

According to these two aspects, DBR is more connected to STA and CC than to NS, which is different in both aspects. CC and DBR are both using true class labels for learning label dependence, but the corresponding base learners are trained on different label subsets: all labels in the case of DBR and only the preceding ones (in the order of the chain) in CC. On the other side, DBR is similar to STA because both are built on top of another multi-label classifier (namely BR in our case) in order to obtain label estimations. The difference is that STA tries to capture labels correlations using such estimations while DBR employs true labels to induce dependent models.

In terms of computational complexity, it is worth noting that all methods allow for learning the base classifiers individually and, therefore, can easily benefit from parallel implementation techniques. Moreover, while the complexity of all approaches scales linearly in the number of labels, they differ in the concrete number of models that are produced: m for CC and NS vs. $2m$ for STA and DBR (m being the number of labels).

4.4. Expanding the feature space: True vs. predicted labels

We conclude this section with a closer look at the first criterion we proposed to distinguish the methods, namely the type of data used in the training phase, and a more in-depth discussion of possible advantages and disadvantages of the two options (true label information vs. predicted).

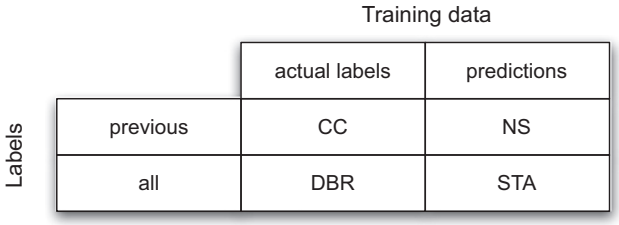

Fig. 2. Properties of chaining and stacking methods.

To begin with, recall that the learning processes of CC and DBR both violate a key assumption of machine learning, namely the assumption that the training data is representative of the test data in the sense of being identically distributed. This assumption does not hold because true label data y_i are used as input attributes during the training phase, while this information is replaced by estimations $\hat{y}_i$ at prediction time. Unless the predictions are perfect, the distribution of the $\hat{y}_i$ will not be identical to the distribution of the y_i. This same problem was previously studied in the context of sequential learning [30]. From the point of view of each binary classifier h_j, this issue can be seen as a problem of attribute noise. More specifically, we are facing the "clean training data vs. noisy test data" case, which is one of the four possible noise scenarios that have been studied in [28].

The question that arises is if using true additional label information during training, which is exactly what CC and DBR seek to exploit in order to perform better than BR, could perhaps be harmful rather than useful. This question is difficult to answer in general, since, as explained in more detail in [22], there are several factors involved. The following two are especially important:

- *The accuracy of label estimations*: The level of attribute noise is in direct correspondence with the accuracy of the estimations used at prediction time. If they are perfect, then the training distribution equals the test distribution, and there is no problem.[3] Otherwise, however, the distributions will differ.
- *The dependency among labels*: Perhaps most interestingly, a (strong enough) dependence between labels is a prerequisite for both, an improvement and a deterioration in comparison to BR. In fact, CC and DBR cannot gain (compared to BR) in case of no label dependence. In that case, however, they are also unlikely to lose, because the binary classifier will most likely ignore the attributes containing label information. On the contrary, in case of pronounced conditional label dependence, they will fully rely on these attributes, and whether or not this is advantageous will mostly depend on the previous factor.

DBR must be less robust against attribute noise than CC, simply because it uses more label estimations. For classifier h_j in a CC model, the larger the number $j-1$ of preceding classifiers in the chain is, the higher is the potential level of attribute noise for h_j. For example, if prediction errors occur independent of each other with probability ε, then the probability of a noise-free input is only $(1-\varepsilon)^{j-1}$. In a DBR model, this value is $(1-\varepsilon)^{m-1}$ for all binary classifiers. Another difference between both is that in a CC model the prediction for each label is used as an estimation for the next labels. Thus, more realistically, one may assume that the probability of a mistake is not constant but will increase with the level of attribute noise in the input [22]. Then, due to the recursive structure of CC, the probability of a mistake will be reinforced and increase even more rapidly along the chain. The order of the chain plays a role, too, since some labels might be inherently more difficult to predict than others. In particular, it would be advantageous to put easier labels in the beginning and harder ones more toward the end of the chain. Notice that DBR is free of this problem, because all label estimations used come from another multi-label classifier, and not from their own predictions.

[3] Strictly speaking, equality does not even hold in this case, since a classifier will normally produce a deterministic prediction, that is, a 0/1-valued probability distribution, even if the true (conditional) distribution is not of that kind.

STA and NS do not suffer from the problem of attribute noise. Training and test data originate from the same source, both are label estimations produced by the same classifiers, so they are identically distributed. However, considering their abilities to detect conditional label dependence, the potential of STA and NS is clearly limited: Since none of these methods ever gets to see a true label y_i as input information during the training phase, there is no way to learn the true label dependence. What they can learn, instead, is the dependency between the predictions $\hat{y}_i$, which can be seen as a noisy version of the true label dependence. Thus, somewhat simplified, one may conclude that CC and DBR are able to learn true label dependence that will only hold approximately at prediction time, whereas STA and NS can only learn a proxy of the label dependence that will nevertheless hold exactly at prediction time.

As already mentioned, the practical performance of all these methods will depend on the level of attribute noise and conditional dependence of the domain. For datasets on which BR already performs strongly—the original input space provides enough information to make accurate predictions—or label dependence is low, the performance of all methods will be similar to that of BR. For those domains with more conditional label dependence and a reasonably good accuracy, CC and DBR are expected to gain in performance compared to BR, STA and NS. Finally, if the accuracy of BR is low and the label dependence high, CC and DBR may perform poorly. In the next section, we will conduct a series of experiments to confirm these claims.

5. Experiments

This section reports the settings and the results of the experiments performed. The main goal of these experiments is to make an exhaustive comparison between stacking and chaining approaches, including our newly proposed one. The aim is to prove which one performs better than the rest for each of the metrics considered, and if one of them is superior with regard to capturing label dependence. For a comparison of some of these methods, for instance CC, with other multi-label learners, see [31].

The second objective is to analyze the behavior of CC and DBR, especially with respect to the attribute noise problem discussed above. Several experiments were carried out to that end, showing that this is a key issue for exploiting label dependence.

5.1. Settings

Before discussing the experimental results, let us describe the common settings of all experiments: datasets, learning algorithms, parameter selection procedures and evaluation metrics.

The experiments were performed over several benchmark multi-label datasets whose main properties are shown in Table 1. As can be seen, there are significant differences in the number of attributes, examples and labels. The cardinality—number of labels per example—varies between 1.07 and 4.24.

The learning algorithms compared were those discussed along the paper: CC, STA, NS and our proposal, DBR. We also included BR as the baseline to show the performance under the label independence assumption. The base learner employed to obtain the binary classifiers that compose all these multi-label models was *logistic regression* [32]. The regularization parameter C was established for each *individual* binary classifier performing a grid search over the values $C \in \{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$ optimizing the accuracy estimated by means of a balanced 2-fold cross validation repeated 5 times. This guarantees that the binary classifiers for a particular label of the compared methods are exactly the same when their respective feature spaces coincide. This happens, for instance, for the first classifier of CC and NS. It will be also equal to that of BR for that label. Another example is the classifier of the last label in a CC model, it will be identical to the classifier of that label in a DBR model. Unlike [11], we do not apply any threshold selection method, and instead we use $t=0.5$ for deciding the relevance of a label in all cases. In fact, our goal is to study the behavior of all approaches without the influence of any factor that may bias the results.

Table 1
Properties of the datasets used in the experiments.

Dataset	Attributes	Examples	Labels	Cardinality
Bibtex	1836	7395	159	2.40
Emotions	72	593	6	1.87
Enron	1001	1702	53	3.38
Genbase	1185	662	27	1.25
Image	135	2000	5	1.24
Mediamill	120	5000	101	4.27
Medical	1449	978	45	1.25
Reuters	243	7119	7	1.24
Scene	294	2407	6	1.07
Slashdot	1079	3782	22	1.18
Yeast	103	2417	14	4.24

The results will be presented in terms of several measures commonly adopted in multi-label classification [12]. Here we consider only example-based evaluation because the goal of the studied classification methods is to detect conditional label dependence. In addition to Hamming and Subset 0/1 loss introduced earlier, we also used the F_1 and Jaccard index, which are defined respectively as follows[4]:

$$F_1(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{2\sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} ([\![y_i = 1]\!] + [\![h_i(\boldsymbol{x}) = 1]\!])}.$$

$$Jaccard(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \frac{\sum_{i=1}^{m} [\![y_i = 1 \text{ and } h_i(\boldsymbol{x}) = 1]\!]}{\sum_{i=1}^{m} [\![y_i = 1 \text{ or } h_i(\boldsymbol{x}) = 1]\!]}.$$

The $F1$ measure is the harmonic mean of precision and recall. Thus, since the number of relevant labels is typically smaller than the number of irrelevant labels, a correct prediction on the former becomes more important than a correct prediction of the latter. This is in contrast to other measures like Hamming, which give the same importance to each individual label. Similar remarks also apply to the Jaccard measure.

As all of the performance measures considered are defined on a per instance basis, the value for a test set is the average over all instances. The scores reported were estimated by means of a 10-fold cross-validation. The rank of a classifier in a particular dataset is indicated in parentheses.[5] The average ranks over all datasets are computed and shown at the last row of each table. Following the recommendations of [33] a two-step comparison for each of the considered measures was performed. The first step consists of a Friedman test of the null hypothesis that states that all approaches perform equally. Then, in the case where this hypothesis is rejected, the Nemenyi test is performed to compare the methods in a pairwise way. Both tests are based on the rank average. The critical differences (CD) in the Nemenyi test depend on the number of datasets and learners compared. They are shown in the caption of each table for different significance levels.

5.2. Experimental results

The first experiment consists of evaluating all methods using benchmark datasets. Tables 2 and 3 show the scores for the

[4] Note that these are accuracy measures instead of loss functions.
[5] Average ranks are shown in case of ties.

Table 2
Experimental results on benchmark datasets for F_1 and Jaccard index. The critical differences for the Nemenyi test are $CD_{p=0.01}=2.19$, $CD_{p=0.05}=1.84$ and $CD_{p=0.1}=1.66$.

Dataset	BR		DBR(BR)		STA		CC		NS	
F1										
Bibtex	0.3702 ± 0.0048	(5.0)	0.3754 ± 0.0059	(2.0)	**0.3781** ± 0.0051	(1.0)	0.3718 ± 0.0051	(4.0)	0.3722 ± 0.0051	(3.0)
Emotions	0.4920 ± 0.0200	(5.0)	**0.6087** ± 0.0140	(1.0)	0.5046 ± 0.0181	(4.0)	0.5893 ± 0.0166	(2.0)	0.5101 ± 0.0147	(3.0)
Enron	0.5566 ± 0.0046	(5.0)	**0.5820** ± 0.0044	(1.0)	0.5585 ± 0.0044	(4.0)	0.5666 ± 0.0057	(2.0)	0.5650 ± 0.0053	(3.0)
Genbase	0.9918 ± 0.0026	(3.5)	**0.9921** ± 0.0026	(1.0)	0.9918 ± 0.0026	(3.5)	0.9918 ± 0.0026	(3.5)	0.9918 ± 0.0026	(3.5)
Image	0.4212 ± 0.0089	(5.0)	**0.5168** ± 0.0104	(1.0)	0.4257 ± 0.0087	(4.0)	0.4628 ± 0.0129	(2.0)	0.4258 ± 0.0090	(3.0)
Mediamill	0.5917 ± 0.0034	(3.0)	0.5723 ± 0.0038	(5.0)	**0.6038** ± 0.0043	(1.0)	0.5859 ± 0.0042	(4.0)	0.6000 ± 0.0032	(2.0)
Medical	0.7733 ± 0.0109	(5.0)	**0.7982** ± 0.0092	(1.0)	0.7840 ± 0.0099	(4.0)	0.7891 ± 0.0095	(2.5)	0.7891 ± 0.0103	(2.5)
Reuters	0.8413 ± 0.0026	(4.5)	**0.8705** ± 0.0023	(1.0)	0.8413 ± 0.0026	(4.5)	0.8610 ± 0.0033	(2.0)	0.8456 ± 0.0031	(3.0)
Scene	0.6125 ± 0.0084	(5.0)	0.6846 ± 0.0082	(2.0)	0.6329 ± 0.0083	(4.0)	**0.6927** ± 0.0093	(1.0)	0.6455 ± 0.0087	(3.0)
Slashdot	0.4433 ± 0.0047	(5.0)	**0.5547** ± 0.0058	(1.0)	0.4448 ± 0.0048	(4.0)	0.5146 ± 0.0069	(2.0)	0.4749 ± 0.0060	(3.0)
Yeast	0.6168 ± 0.0081	(3.0)	0.6098 ± 0.0084	(5.0)	0.6164 ± 0.0081	(4.0)	**0.6264** ± 0.0089	(1.0)	0.6192 ± 0.0074	(2.0)
Avg. rank		(4.45)		(1.91)		(3.45)		(2.36)		(2.82)
Jaccard Index										
Bibtex	0.3150 ± 0.0048	(5.0)	**0.3232** ± 0.0059	(1.0)	0.3217 ± 0.0050	(2.0)	0.3190 ± 0.0051	(3.0)	0.3185 ± 0.0050	(4.0)
Emotions	0.4227 ± 0.0186	(5.0)	**0.5176** ± 0.0136	(1.0)	0.4350 ± 0.0170	(4.0)	0.5149 ± 0.0172	(2.0)	0.4413 ± 0.0150	(3.0)
Enron	0.4469 ± 0.0042	(5.0)	**0.4709** ± 0.0048	(1.0)	0.4491 ± 0.0038	(4.0)	0.4628 ± 0.0059	(2.0)	0.4557 ± 0.0045	(3.0)
Genbase	0.9894 ± 0.0026	(3.5)	**0.9897** ± 0.0026	(1.0)	0.9894 ± 0.0026	(3.5)	0.9894 ± 0.0026	(3.5)	0.9894 ± 0.0026	(3.5)
Image	0.3860 ± 0.0091	(5.0)	**0.4732** ± 0.0096	(1.0)	0.3910 ± 0.0089	(3.0)	0.4286 ± 0.0136	(2.0)	0.3906 ± 0.0092	(4.0)
Mediamill	0.4670 ± 0.0039	(4.0)	0.4542 ± 0.0040	(5.0)	**0.4836** ± 0.0048	(1.0)	0.4710 ± 0.0043	(3.0)	0.4790 ± 0.0037	(2.0)
Medical	0.7451 ± 0.0109	(5.0)	**0.7695** ± 0.0092	(1.0)	0.7556 ± 0.0100	(4.0)	0.7642 ± 0.0092	(2.0)	0.7637 ± 0.0100	(3.0)
Reuters	0.8167 ± 0.0029	(5.0)	**0.8400** ± 0.0026	(1.0)	0.8169 ± 0.0028	(4.0)	0.8364 ± 0.0038	(2.0)	0.8219 ± 0.0034	(3.0)
Scene	0.5941 ± 0.0083	(5.0)	0.6400 ± 0.0080	(2.0)	0.6177 ± 0.0084	(4.0)	**0.6786** ± 0.0093	(1.0)	0.6299 ± 0.0087	(3.0)
Slashdot	0.4271 ± 0.0047	(5.0)	**0.4970** ± 0.0060	(1.0)	0.4286 ± 0.0048	(4.0)	0.4942 ± 0.0065	(2.0)	0.4570 ± 0.0056	(3.0)
Yeast	0.5071 ± 0.0085	(3.0)	0.4968 ± 0.0090	(5.0)	0.5070 ± 0.0086	(4.0)	**0.5231** ± 0.0098	(1.0)	0.5110 ± 0.0084	(2.0)
Avg. rank		(4.59)		(1.82)		(3.41)		(2.14)		(3.05)

Table 3
Experimental results on benchmark datasets for Hamming loss and Subset 0/1 loss. The critical differences for the Nemenyi test are $CD_{p=0.01}=2.19$, $CD_{p=0.05}=1.84$ and $CD_{p=0.1}=1.66$.

Dataset	BR		DBR(BR)		STA		CC		NS	
Hamming Loss										
Bibtex	**0.0121** ± 0.0001	(2.5)	**0.0121** ± 0.0001	(2.5)	0.0122 ± 0.0001	(5.0)	**0.0121** ± 0.0001	(2.5)	**0.0121** ± 0.0001	(2.5)
Emotions	0.2203 ± 0.0067	(4.0)	0.2315 ± 0.0087	(5.0)	0.2184 ± 0.0061	(3.0)	**0.2164** ± 0.0081	(1.0)	0.2175 ± 0.0054	(2.0)
Enron	0.0446 ± 0.0007	(2.5)	0.0488 ± 0.0006	(5.0)	0.0446 ± 0.0007	(2.5)	0.0464 ± 0.0008	(4.0)	**0.0445** ± 0.0006	(1.0)
Genbase	0.0008 ± 0.0001	(3.5)	**0.0007** ± 0.0001	(1.0)	0.0008 ± 0.0001	(3.5)	0.0008 ± 0.0001	(3.5)	0.0008 ± 0.0001	(3.5)
Image	0.2025 ± 0.0041	(4.0)	0.2161 ± 0.0067	(5.0)	0.2018 ± 0.0041	(3.0)	**0.2013** ± 0.0051	(1.0)	0.2017 ± 0.0043	(2.0)
Mediamill	0.0276 ± 0.0003	(3.0)	0.0310 ± 0.0004	(5.0)	**0.0269** ± 0.0003	(1.0)	0.0286 ± 0.0003	(4.0)	0.0271 ± 0.0003	(2.0)
Medical	0.0099 ± 0.0004	(4.5)	**0.0098** ± 0.0004	(1.5)	0.0098 ± 0.0004	(3.0)	0.0099 ± 0.0004	(4.5)	**0.0098** ± 0.0004	(1.5)
Reuters	0.0458 ± 0.0008	(3.0)	0.0577 ± 0.0012	(5.0)	0.0457 ± 0.0008	(2.0)	0.0476 ± 0.0010	(4.0)	**0.0457** ± 0.0008	(1.0)
Scene	0.0983 ± 0.0029	(3.0)	0.1831 ± 0.0047	(5.0)	**0.0935** ± 0.0028	(1.0)	0.1069 ± 0.0030	(4.0)	0.0970 ± 0.0030	(2.0)
Slashdot	**0.0373** ± 0.0004	(1.0)	0.0880 ± 0.0019	(5.0)	0.0374 ± 0.0005	(2.0)	0.0402 ± 0.0006	(4.0)	0.0379 ± 0.0005	(3.0)
Yeast	0.1981 ± 0.0032	(2.5)	0.2156 ± 0.0036	(5.0)	0.1981 ± 0.0033	(2.5)	0.2099 ± 0.0041	(4.0)	**0.1978** ± 0.0031	(1.0)
Avg. rank		(3.05)		(4.09)		(2.59)		(3.32)		(1.95)
Subset 0/1 Loss										
Bibtex	0.8283 ± 0.0043	(5.0)	**0.8154** ± 0.0053	(1.0)	0.8269 ± 0.0046	(4.0)	0.8211 ± 0.0049	(2.0)	0.8226 ± 0.0044	(3.0)
Emotions	0.7942 ± 0.0156	(5.0)	0.7470 ± 0.0174	(2.0)	0.7825 ± 0.0158	(4.0)	**0.7098** ± 0.0216	(1.0)	0.7741 ± 0.0192	(3.0)
Enron	0.8690 ± 0.0055	(5.0)	0.8525 ± 0.0088	(2.0)	0.8661 ± 0.0054	(4.0)	**0.8408** ± 0.0070	(1.0)	0.8578 ± 0.0053	(3.0)
Genbase	**0.0181** ± 0.0030	(3.0)	**0.0181** ± 0.0030	(3.0)	**0.0181** ± 0.0030	(3.0)	**0.0181** ± 0.0030	(3.0)	**0.0181** ± 0.0030	(3.0)
Image	0.7150 ± 0.0097	(5.0)	**0.6520** ± 0.0133	(1.0)	0.7095 ± 0.0101	(3.0)	0.6700 ± 0.0161	(2.0)	0.7110 ± 0.0102	(4.0)
Mediamill	0.9036 ± 0.0047	(5.0)	0.8814 ± 0.0039	(3.0)	0.8794 ± 0.0051	(2.0)	**0.8620** ± 0.0043	(1.0)	0.8834 ± 0.0042	(4.0)
Medical	0.3394 ± 0.0123	(5.0)	0.3149 ± 0.0094	(3.0)	0.3292 ± 0.0115	(4.0)	**0.3098** ± 0.0098	(1.0)	0.3118 ± 0.0103	(2.0)
Reuters	0.2569 ± 0.0043	(5.0)	0.2460 ± 0.0042	(2.0)	0.2561 ± 0.0043	(4.0)	**0.2375** ± 0.0056	(1.0)	0.2484 ± 0.0049	(3.0)
Scene	0.4603 ± 0.0100	(5.0)	0.4499 ± 0.0085	(4.0)	0.4275 ± 0.0098	(3.0)	**0.3635** ± 0.0097	(1.0)	0.4167 ± 0.0097	(2.0)
Slashdot	0.6195 ± 0.0055	(4.0)	0.6285 ± 0.0063	(5.0)	0.6182 ± 0.0055	(3.0)	**0.5651** ± 0.0065	(1.0)	0.5949 ± 0.0056	(2.0)
Yeast	0.8453 ± 0.0100	(5.0)	0.8407 ± 0.0120	(3.0)	0.8432 ± 0.0099	(4.0)	**0.7865** ± 0.0119	(1.0)	0.8246 ± 0.0115	(2.0)
Avg. rank		(4.73)		(2.64)		(3.45)		(1.36)		(2.82)

evaluation metrics. For DBR, the required label estimations are provided by a BR classifier, hence it is denoted DBR(BR).

Interestingly, all the stacking and chaining approaches outperform BR in F_1, Jaccard index and Subset 0/1 loss. This is not surprising, because BR is well-tailored for Hamming loss, but not for the other metrics. However, it is worth noting that STA and NS obtain a better ranking than BR in all measures, including Hamming loss, but their differences are only significant in the case of NS for Subset 0/1 loss.

DBR(BR) is the best method for F_1 measure, followed by CC. Both are significantly better than BR, at level $p<0.01$ for DBR(BR) and $p<0.05$ for CC. The rest of the differences are not significant. Despite STA and NS also boost the scores of BR, showing that some

improvements can be obtained applying the stacking idea, these results seem to suggest that inducing multi-label models using actual label information, as CC and DBR do, helps to better predict the relevant labels.

In terms of Jaccard index, which is closely related to F_1, the results are quite similar. DBR(BR) is the best method and CC is the second one; both are significantly better than BR at level $p < 0.01$. NS and STA compare unfavorably with their respective counterparts, showing again that using predictions is worse to capture the dependency among labels. BR is again the worst method, confirming that its label independence assumption is less appropriate for this sort of performance measures.

The results for Hamming loss are quite interesting. NS obtains the best scores, followed by STA and BR; the differences between them are rather small. Nevertheless, it may surprise that NS and STA improve the average rank of BR, which optimizes Hamming loss given the original input space. Our explanation for this result is that (i) these methods use an extended feature space, in which they are able to detect some label correlations, and correct the predictions of BR in the case of STA (remember that STA use the outputs of BR as input features, including the prediction for the own label), and (ii) unlike CC and DBR, they do not suffer the attribute noise problem discussed above. We will study this aspect in more detail in the next experiment. For all these reasons, NS, STA and BR should produce similar predictions. On the other hand, DBR(BR) and CC are both outperformed by BR and stacking approaches, being DBR(BR) the worst algorithm: it is beaten significantly by NS ($p < 0.05$).

Finally and as it was expected, CC outperforms the rest of the methods in Subset 0/1 loss. CC is significantly better than STA ($p < 0.05$) and BR ($p < 0.01$), which in turn is also significantly worse than DBR(BR) and NS ($p < 0.05$). Note that the chaining approaches, CC and NS, based on the product rule of probability, perform better than their respective counterparts considering the kind of training data used: CC vs. DBR(BR) (true labels), and NS vs. STA (predictions). However, note that those methods which use true labels during training perform better than those using predictions.

The bottom line of these results is that we have studied three different types of loss functions and in each of them one kind of method is better: F_1 and Jaccard favoring those predictions that include the relevant labels (DBR(BR)), Hamming loss needs good marginal estimations (NS), and Subset 0/1 loss requires to estimate the joint distribution (CC). The other conclusion is that the first criterion (using true labels vs. predictions as additional training information) has a stronger influence on the final behavior of the learner than the second one (partial vs. full conditioning). It seems that using predicted labels is better for reducing the Hamming loss, while true labels are advantageous for measures that are biased to the relevant labels.

5.3. Analyzing the problem of attribute noise

We have previously stated that, in principle, DBR should perform well in terms of Hamming loss considering the augmented feature space formed by the original input space and the label space. However, the disappointing results achieved by DBR in Hamming loss do not seem to support our claim. The most probable reason is the attribute noise in the feature space used. To prove this conjecture, we carried out several experiments.

The first one is quite simple: the idea was to completely remove this attribute noise in the two methods that are affected by this problem: CC and DBR. To that end, when the prediction $h_j(\boldsymbol{x})$ is calculated, instead of using the estimations of the other labels as usual, those values are replaced by the true labels. For instance, if h_j is one of the binary classifiers of a DBR model, its feature space is augmented with the actual values of the rest of the labels: $y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_m$; notice that the true label of y_j is not used. One may argue that this experiment will produce unrealistically good results, and that we are "cheating" in the test phase. Although this is perfectly true, the experiments will hopefully help us in finding answers to several questions: Can these learners really detect label dependence? What are the differences between CC and DBR in this regard? To what extent could BR results be improved? To what extent are CC and DBR affected by attribute noise problem?

Table 4 reports the scores of the experiment for Hamming loss. In addition to the results of CC and DBR using perfect label information in the test phase, denoted as DBR(True Labels) and CC(True Labels) respectively, the scores of DBR(BR) and CC of the previous experiment (Table 3) are included. For all these learners, Table 4 shows the improvement with respect to BR, which is used as a reference.

On an average, CC ameliorates the results of BR in more than 15%, while the improvement of DBR is 35%. Thus, the first conclusion is that both methods can detect significant label dependencies. In that sense, it is worth noting that DBR uses all the information available, therefore it comes at no surprise that DBR models are able to detect more relationships between labels. The first classifiers in the chain of a CC model are usually quite similar to those of BR: the first classifier is equal, and the next ones can be alike whenever the labels placed at the beginning of the chain do not present much correlation. The improvement is mainly due to the classifiers at the end of the chain. Contrarily, all binary classifiers of a DBR model should be better than their BR counterparts, simply because they are using more information, without any noise in this experiment.

However, this advantage of DBR comes at a cost. DBR is much more sensitive to the attribute noise problem. This can be

Table 4
Experimental results of CC and DBR when both use true label information (in a cheating way) during the testing phase. The table includes the scores of BR, DBR(BR) and CC previously reported in Table 3. The improvements with respect to BR results are shown as percentages for all methods.

Dataset	BR	DBR(BR)	Im. (%)	DBR(TL)	Im. (%)	CC	Im. (%)	CC(TL)	Im. (%)
Hamming Loss									
Bibtex	0.0121 ± 0.0001	0.0121 ± 0.0001	0.0	0.0113 ± 0.0001	6.6	0.0121 ± 0.0001	−0.3	0.0117 ± 0.0001	3.7
Emotions	0.2203 ± 0.0067	0.2315 ± 0.0087	−5.1	0.1535 ± 0.0075	30.3	0.2164 ± 0.0081	1.8	0.1841 ± 0.0072	16.4
Enron	0.0446 ± 0.0007	0.0488 ± 0.0006	−9.4	0.0374 ± 0.0006	16.1	0.0464 ± 0.0008	−4.1	0.0410 ± 0.0007	8.1
Genbase	0.0008 ± 0.0001	0.0007 ± 0.0001	12.5	0.0007 ± 0.0001	12.5	0.0008 ± 0.0001	2.0	0.0007 ± 0.0001	9.0
Image	0.2025 ± 0.0041	0.2161 ± 0.0067	−6.7	0.1623 ± 0.0054	19.9	0.2013 ± 0.0051	0.6	0.1906 ± 0.0051	5.9
Mediamill	0.0276 ± 0.0003	0.0310 ± 0.0004	−12.3	0.0142 ± 0.0002	48.6	0.0286 ± 0.0003	−3.8	0.0203 ± 0.0002	26.5
Medical	0.0099 ± 0.0004	0.0098 ± 0.0004	1.0	0.0068 ± 0.0004	31.3	0.0099 ± 0.0004	0.2	0.0085 ± 0.0004	14.4
Reuters	0.0458 ± 0.0008	0.0577 ± 0.0012	−26.0	0.0318 ± 0.0008	30.6	0.0476 ± 0.0010	−3.9	0.0406 ± 0.0010	11.3
Scene	0.0983 ± 0.0029	0.1831 ± 0.0047	−86.3	0.0238 ± 0.0012	75.8	0.1069 ± 0.0030	−8.8	0.0724 ± 0.0026	26.4
Slashdot	0.0373 ± 0.0004	0.0880 ± 0.0019	−135.9	0.0262 ± 0.0006	29.8	0.0402 ± 0.0006	−7.9	0.0340 ± 0.0005	8.9
Yeast	0.1981 ± 0.0032	0.2156 ± 0.0036	−8.8	0.0331 ± 0.0011	83.3	0.2099 ± 0.0041	−6.0	0.1170 ± 0.0019	40.9
Avg.			−25.2		35.0		−2.7		15.6

observed with the negative improvements of DBR(BR), which are bigger than those of regular CC. Our explanation for this fact is that the binary classifiers of DBR seem to intensively use the information coming from the rest of the labels, which is in principle good because they are detecting those relationships, but can have a harmful impact when the estimation of some labels is poor. This is especially dangerous with a measure like Hamming loss, in which every error counts in the same proportion. Measuring the outcomes for the other performance metrics, we observed that the improvements of DBR(True Labels) are in fact smaller, but DBR(BR) still enhances the scores of BR, as can be seen in Tables 2 and 3.

Another consequence of the experiment is that it proves that exploiting conditional dependence could help to boost the performance in Hamming loss. It is well known that BR optimizes such measure given the original input space, but new learners can be devised to properly learn from the information contained in the label space, improving the performance for Hamming loss. These new methods must solve some potential issues, like the attribute noise discussed here or others, including the complex dependences (e.g. cycles) that labels could present.

Finally, the experiment shows how DBR can be useful to observe the level of label dependence in multi-label domains. For instance, some datasets (e.g. bibtex) do not display much label dependence, while others like mediamill, scene or yeast do. The improvement in these domains is quite impressive, suggesting that the label space contains valuable information that should be exploited.

We performed a second experiment in order to deeply study this noise problem. The goal was to show the performance of DBR under different levels of attribute noise. Several 3D-graphs were built for this purpose, one per each evaluation measure and per dataset.[6] In these graphs (see Figs. 3 and 4), the scores are drawn as a function of two parameters, namely the percentages of false positives (FP) and true positives (TP) in the label estimations used during the test phase of DBR. The first dimension was ranged from 0% to 10%, whereas the second one varies between 0% and 100%. The election of 10% in case of the FP is due to the sparseness of the label space. For each pair of values, 30 different label estimations were drawn randomly and used by the same DBR model to output its predictions. The graphs show the average for each combination. For the loss functions, Hamming and Subset 0/1, figures respectively show 1-Hamming Loss and 1-Subset 0/1 loss; thus for all measures, the higher the better. Besides, the hyperplane in the graphs corresponds to the BR score, representing the performance under label independence assumption. Notice that the scale of Z-axis is different among graphics.

Looking at the figures, some steady behaviors can be observed for all performance measures and datasets. Firstly, the inclusion of TP in the label space to feed DBR models rapidly improves the performance. Although it is true that this increase in performance begins to be not so pronounced as FP are included. The improvement degree attained with respect to BR performance depends not only on the dataset, but also on the measure. In fact, F_1 results tend to be the scores that can be more easily enhanced, followed by Jaccard index and Subset 0/1 loss. In the other extreme we find Hamming loss whose results are hard to be improved on several datasets. Studying the figures for all benchmark datasets, there are some domains where more FP hardly worsens the performance independent of the measure. Besides, it also happens that for the same domain the performance keeps equal for some measures and it worsens for others. That is the case of emotions dataset (see Fig. 3, top) where the influence on F_1 scores is remarkable with regard to the others, or the case of medical dataset (Fig. 3, bottom) for F_1 and Jaccard index. In general, we can state that F_1 and Jaccard index are more sensitive as FP are added than in case of Hamming loss or Subset 0/1 loss, which are more affected by the percentage of TP in the label estimations.

The figures included here try to exemplify quite different behaviors of DBR depending on the dataset and on the performance measures. Fig. 3 (top) shows the graphics for emotions dataset. Let us remember that DBR(BR) improves the scores of BR for this domain in terms of F_1, Jaccard index and Subset 0/1 loss (see Tables 2 and 3). These results can be explained with this experiment. Firstly, F_1 and Jaccard index scores attained by BR are improved by DBR with almost any combination of TP and FP values. Interestingly, DBR performs better without any information about the rest of the labels (TP=0 and FP=0), showing that it is able to learn better models for these measures. Instead, it is not unlikely that DBR obtains a worse result for Hamming loss, especially when the label estimations used contain several mistakes, as it happens with DBR(BR), given that the percentage of misclassified labels (Hamming loss) of BR is greater than 22%, the highest one among all datasets. Finally, DBR seems a little bit more robust against attribute noise for Subset 0/1 loss.

In our first experiment in Section 5.2, the only domain in which DBR(BR) ameliorates the results of BR in all measures, including Hamming loss, was medical dataset. Looking at its graphs (Fig. 3, bottom), we can observe that it is relatively easy to improve BR's results in all metrics: only more than 50% of TP is needed, alongside with a not very large number of FP. Taken into account that the score of BR for Hamming loss is the second lowest, less than 1% of mistakes (see Table 3), the attribute noise problem is not a big issue on this domain, allowing DBR to obtain better scores.

A quite interesting example is provided by scene dataset (Fig. 4, top). According to Tables 2 and 3, DBR(BR) improves the scores of BR, except for Hamming loss. The graphics show that this measure is the most sensitive to noise, DBR would need more than 80% of TP, while for the rest of the measures this number could be much lower, around 55% and sometimes less, if the percentage of FP is moderate.

Finally, graphics of yeast dataset in Fig. 4 (bottom) present a case in which DBR has severe difficulties to enhance BR's results in all measures. In fact, DBR(BR) only obtains better scores than BR for Subset 0/1 loss and the difference is low. The figures for all measures are quite similar, requiring a high percentage of TP to improve over BR (notice that FPs hardly have influence). The problem, again, is that the Hamming loss of BR is large, more than 19% and those mistakes introduce noise in the predictions of DBR (BR), which in turn produce even more mistakes. The pity is that the improvements could be amazing in this domain, more than 20 points for each metric.

The main conclusion of this experiment has to do with the influence of attribute noise with respect to the performance measures. Hamming loss is by far the metric more sensitive. Our explanation for this fact is that for Hamming loss every mistake has the same influence, but for others metrics, like F_1, this is not the case. For instance, if all relevant labels of an instance are correctly predicted, the F_1 will decrease in less quantity with each irrelevant label predicted as relevant. This suggests that DBR tends to predict some irrelevant labels as relevant, which is not so harmful for measures like F_1.

Furthermore, the graphs provide us more information than just the behavior of our method. The inclusion of the BR performance in the figures gives us an idea about the dependence among labels that exists in the datasets. In this way, we would be able to carry out a classification of the datasets according to it.

5.4. Using other multi-label classifiers to obtain label estimations

The experiments of the previous section prove that DBR heavily depends on the label estimations used during the test phase.

[6] All of them are included in the supplementary material of the paper.

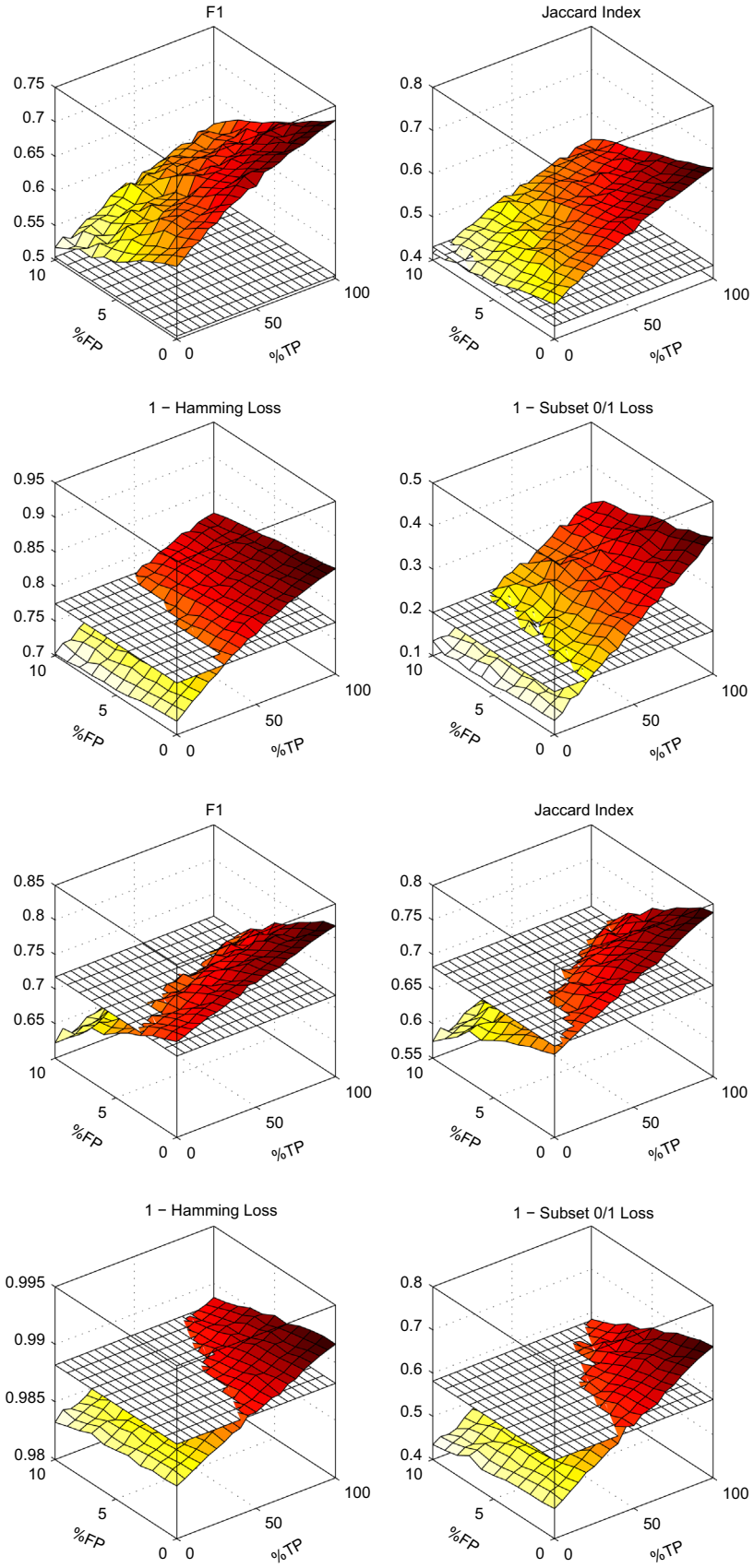

Fig. 3. F_1, Jaccard index, Hamming loss and Subset 0/1 loss for emotions (top) and medical (bottom) datasets.

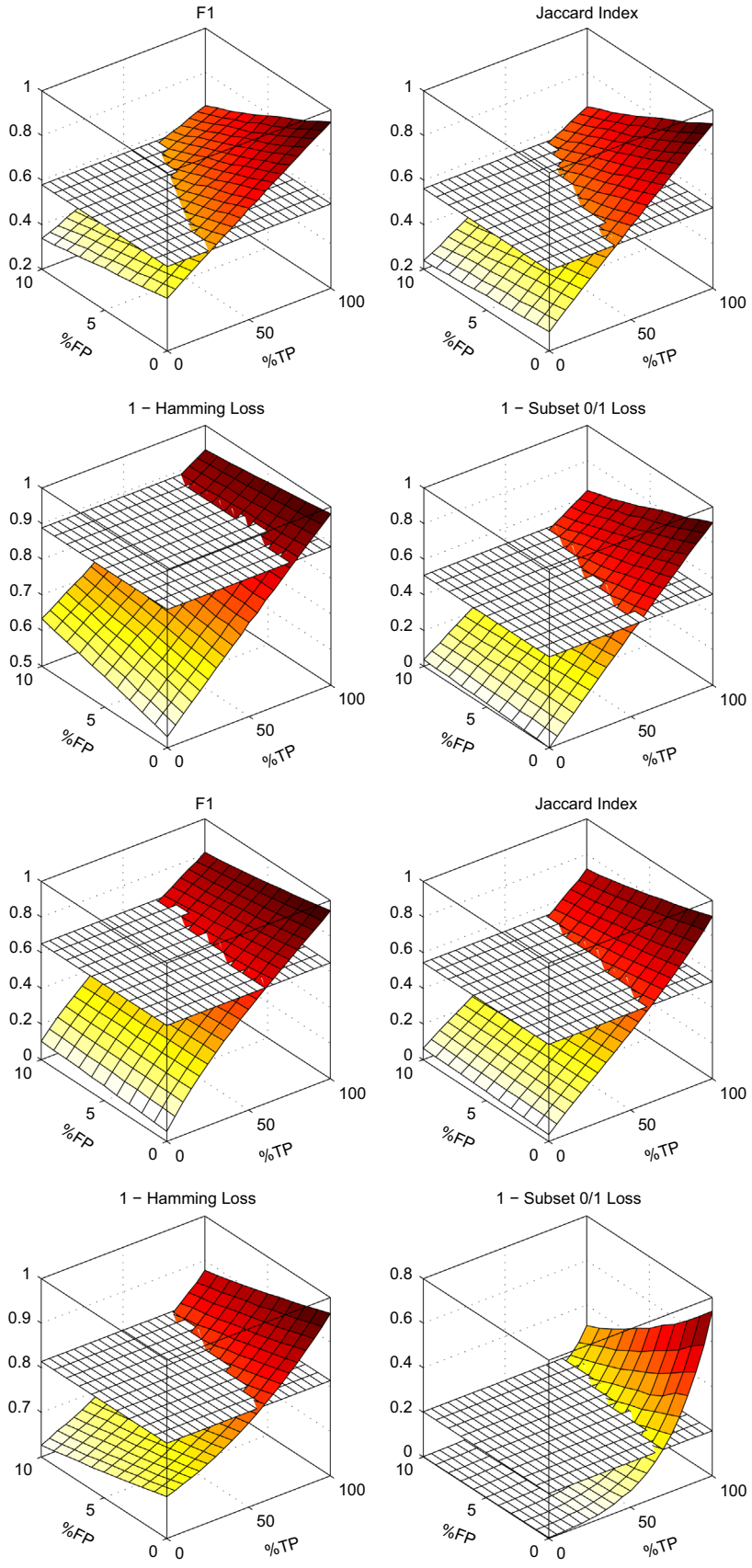

Fig. 4. F_1, Jaccard index, Hamming loss and Subset 0/1 loss for scene (top) and yeast (bottom) datasets.

Table 5
Performance of DBR, in terms of F_1 and Jaccard index, when different multi-label classifiers are used to provide the required labels estimations. The scores of these multi-label classifiers are included for reference.

Dataset	BR		DBR(BR)		STA		DBR(STA)		CC		DBR(CC)		NS		DBR(NS)	
*F*1																
Bibtex	0.3702 ± 0.0048	(8.0)	0.3754 ± 0.0059	(3.5)	**0.3781** ± 0.0051	(1.0)	0.3769 ± 0.0060	(2.0)	0.3718 ± 0.0051	(7.0)	0.3744 ± 0.0061	(5.0)	0.3722 ± 0.0051	(6.0)	0.3754 ± 0.0058	(3.5)
Emotions	0.4920 ± 0.0200	(8.0)	0.6087 ± 0.0140	(3.0)	0.5046 ± 0.0181	(7.0)	**0.6110** ± 0.0123	(1.0)	0.5893 ± 0.0166	(5.0)	0.6042 ± 0.0149	(4.0)	0.5101 ± 0.0147	(6.0)	0.6092 ± 0.0124	(2.0)
Enron	0.5566 ± 0.0046	(8.0)	0.5820 ± 0.0044	(3.0)	0.5585 ± 0.0044	(7.0)	0.5822 ± 0.0043	(2.0)	0.5666 ± 0.0057	(5.0)	0.5786 ± 0.0071	(4.0)	0.5650 ± 0.0053	(6.0)	**0.5842** ± 0.0039	(1.0)
Genbase	0.9918 ± 0.0026	(6.5)	**0.9921** ± 0.0026	(2.5)	0.9918 ± 0.0026	(6.5)	**0.9921** ± 0.0026	(2.5)	0.9918 ± 0.0026	(6.5)	**0.9921** ± 0.0026	(2.5)	0.9918 ± 0.0026	(6.5)	**0.9921** ± 0.0026	(2.5)
Mediamill	0.5917 ± 0.0034	(3.0)	0.5723 ± 0.0038	(8.0)	**0.6038** ± 0.0043	(1.0)	0.5885 ± 0.0047	(4.0)	0.5859 ± 0.0042	(5.0)	0.5765 ± 0.0039	(7.0)	0.6000 ± 0.0032	(2.0)	0.5837 ± 0.0038	(6.0)
Medical	0.7733 ± 0.0109	(8.0)	0.7982 ± 0.0092	(2.0)	0.7840 ± 0.0099	(7.0)	0.7968 ± 0.0093	(4.0)	0.7891 ± 0.0095	(5.5)	**0.8005** ± 0.0092	(1.0)	0.7891 ± 0.0103	(5.5)	0.7975 ± 0.0093	(3.0)
Image	0.4212 ± 0.0089	(8.0)	0.5168 ± 0.0104	(2.0)	0.4257 ± 0.0087	(7.0)	**0.5169** ± 0.0117	(1.0)	0.4628 ± 0.0129	(5.0)	0.5127 ± 0.0093	(4.0)	0.4258 ± 0.0090	(6.0)	0.5158 ± 0.0111	(3.0)
Reuters	0.8413 ± 0.0026	(7.5)	0.8705 ± 0.0023	(4.0)	0.8413 ± 0.0026	(7.5)	0.8710 ± 0.0024	(3.0)	0.8610 ± 0.0033	(5.0)	0.8727 ± 0.0031	(2.0)	0.8456 ± 0.0031	(6.0)	**0.8750** ± 0.0028	(1.0)
Scene	0.6125 ± 0.0084	(8.0)	0.6846 ± 0.0082	(5.0)	0.6329 ± 0.0083	(7.0)	**0.7057** ± 0.0084	(1.0)	0.6927 ± 0.0093	(4.0)	0.6961 ± 0.0084	(3.0)	0.6455 ± 0.0087	(6.0)	0.7027 ± 0.0093	(2.0)
Slashdot	0.4433 ± 0.0047	(8.0)	0.5547 ± 0.0058	(4.0)	0.4448 ± 0.0048	(7.0)	0.5555 ± 0.0059	(3.0)	0.5146 ± 0.0069	(5.0)	0.5608 ± 0.0074	(2.0)	0.4749 ± 0.0060	(6.0)	**0.5613** ± 0.0069	(1.0)
Yeast	0.6168 ± 0.0081	(4.0)	0.6098 ± 0.0084	(8.0)	0.6164 ± 0.0081	(5.0)	0.6107 ± 0.0082	(7.0)	**0.6264** ± 0.0089	(1.0)	0.6260 ± 0.0090	(2.0)	0.6192 ± 0.0074	(3.0)	0.6147 ± 0.0076	(6.0)
Avg. rank		(7.00)		(4.09)		(5.73)		(2.77)		(4.91)		(3.32)		(5.36)		(2.82)
Jaccard Index																
Bibtex	0.3150 ± 0.0048	(8.0)	0.3232 ± 0.0059	(3.0)	0.3217 ± 0.0050	(5.0)	**0.3245** ± 0.0060	(1.0)	0.3190 ± 0.0051	(6.0)	0.3229 ± 0.0062	(4.0)	0.3185 ± 0.0050	(7.0)	0.3233 ± 0.0059	(2.0)
Emotions	0.4227 ± 0.0186	(8.0)	0.5176 ± 0.0136	(4.0)	0.4350 ± 0.0170	(7.0)	0.5212 ± 0.0118	(3.0)	0.5149 ± 0.0172	(5.0)	**0.5255** ± 0.0151	(1.0)	0.4413 ± 0.0150	(6.0)	0.5216 ± 0.0115	(2.0)
Enron	0.4469 ± 0.0042	(8.0)	0.4709 ± 0.0048	(4.0)	0.4491 ± 0.0038	(7.0)	0.4712 ± 0.0046	(3.0)	0.4628 ± 0.0059	(5.0)	0.4739 ± 0.0070	(2.0)	0.4557 ± 0.0045	(6.0)	**0.4742** ± 0.0044	(1.0)
Genbase	0.9894 ± 0.0026	(6.5)	**0.9897** ± 0.0026	(2.5)	0.9894 ± 0.0026	(6.5)	**0.9897** ± 0.0026	(2.5)	0.9894 ± 0.0026	(6.5)	**0.9897** ± 0.0026	(2.5)	0.9894 ± 0.0026	(6.5)	**0.9897** ± 0.0026	(2.5)
Mediamill	0.4670 ± 0.0039	(6.0)	0.4542 ± 0.0040	(8.0)	**0.4836** ± 0.0048	(1.0)	0.4749 ± 0.0052	(3.0)	0.4710 ± 0.0043	(4.0)	0.4647 ± 0.0039	(7.0)	0.4790 ± 0.0037	(2.0)	0.4699 ± 0.0040	(5.0)
Medical	0.7451 ± 0.0109	(8.0)	0.7695 ± 0.0092	(4.0)	0.7556 ± 0.0100	(7.0)	0.7706 ± 0.0090	(3.0)	0.7642 ± 0.0092	(5.0)	**0.7756** ± 0.0092	(1.0)	0.7637 ± 0.0100	(6.0)	0.7724 ± 0.0092	(2.0)
Image	0.3860 ± 0.0091	(8.0)	0.4732 ± 0.0096	(2.0)	0.3910 ± 0.0089	(6.0)	**0.4739** ± 0.0109	(1.0)	0.4286 ± 0.0136	(5.0)	0.4725 ± 0.0092	(3.0)	0.3906 ± 0.0092	(7.0)	0.4723 ± 0.0102	(4.0)
Reuters	0.8167 ± 0.0029	(8.0)	0.8400 ± 0.0026	(4.0)	0.8169 ± 0.0028	(7.0)	0.8404 ± 0.0028	(3.0)	0.8364 ± 0.0038	(5.0)	**0.8478** ± 0.0036	(1.0)	0.8219 ± 0.0034	(6.0)	0.8456 ± 0.0032	(2.0)
Scene	0.5941 ± 0.0083	(8.0)	0.6400 ± 0.0080	(5.0)	0.6177 ± 0.0084	(7.0)	0.6638 ± 0.0081	(4.0)	0.6786 ± 0.0093	(2.0)	**0.6826** ± 0.0083	(1.0)	0.6299 ± 0.0087	(6.0)	0.6673 ± 0.0091	(3.0)
Slashdot	0.4271 ± 0.0047	(8.0)	0.4970 ± 0.0060	(4.0)	0.4286 ± 0.0048	(7.0)	0.4979 ± 0.0061	(3.0)	0.4942 ± 0.0065	(5.0)	**0.5207** ± 0.0074	(1.0)	0.4570 ± 0.0056	(6.0)	0.5101 ± 0.0068	(2.0)
Yeast	0.5071 ± 0.0085	(4.0)	0.4968 ± 0.0090	(8.0)	0.5070 ± 0.0086	(5.0)	0.4981 ± 0.0089	(7.0)	**0.5231** ± 0.0098	(1.0)	0.5228 ± 0.0098	(2.0)	0.5110 ± 0.0084	(3.0)	0.5040 ± 0.0090	(6.0)
Avg. rank		(7.32)		(4.41)		(5.95)		(3.05)		(4.50)		(2.32)		(5.59)		(2.86)

Table 6
Performance of DBR, in terms of Hamming loss and Subset 0/1 loss, when different multi-label classifiers are used to provide the required labels estimations. The scores of these multi-label classifiers are included for reference.

Dataset	BR		DBR(BR)		STA		DBR(STA)		CC		DBR(CC)		NS		DBR(NS)	
Hamming Loss																
Bibtex	**0.0121** ± 0.0001	(3.0)	**0.0121** ± 0.0001	(3.0)	0.0122 ± 0.0001	(7.0)	**0.0121** ± 0.0001	(3.0)	**0.0121** ± 0.0001	(3.0)	0.0122 ± 0.0001	(7.0)	**0.0121** ± 0.0001	(3.0)	0.0122 ± 0.0001	(7.0)
Emotions	0.2203 ± 0.0067	(4.0)	0.2315 ± 0.0087	(8.0)	0.2184 ± 0.0061	(3.0)	0.2310 ± 0.0079	(7.0)	**0.2164** ± 0.0081	(1.0)	0.2254 ± 0.0085	(5.0)	0.2175 ± 0.0054	(2.0)	0.2296 ± 0.0072	(6.0)
Enron	0.0446 ± 0.0007	(2.5)	0.0488 ± 0.0006	(8.0)	0.0446 ± 0.0007	(2.5)	0.0486 ± 0.0006	(6.0)	0.0464 ± 0.0008	(4.0)	0.0476 ± 0.0007	(5.0)	**0.0445** ± 0.0006	(1.0)	0.0487 ± 0.0005	(7.0)
Genbase	0.0008 ± 0.0001	(6.5)	**0.0007** ± 0.0001	(2.5)	0.0008 ± 0.0001	(6.5)	**0.0007** ± 0.0001	(2.5)	0.0008 ± 0.0001	(6.5)	**0.0007** ± 0.0001	(2.5)	0.0008 ± 0.0001	(6.5)	**0.0007** ± 0.0001	(2.5)
Mediamill	0.0276 ± 0.0003	(3.0)	0.0310 ± 0.0004	(8.0)	**0.0269** ± 0.0003	(1.0)	0.0294 ± 0.0004	(6.0)	0.0286 ± 0.0003	(4.0)	0.0292 ± 0.0003	(5.0)	0.0271 ± 0.0003	(2.0)	0.0296 ± 0.0003	(7.0)
Medical	0.0099 ± 0.0004	(7.5)	0.0098 ± 0.0004	(4.5)	0.0098 ± 0.0004	(4.5)	**0.0096** ± 0.0004	(1.0)	0.0099 ± 0.0004	(7.5)	0.0097 ± 0.0004	(2.0)	0.0098 ± 0.0004	(4.5)	0.0098 ± 0.0004	(4.5)
Image	0.2025 ± 0.0041	(4.0)	0.2161 ± 0.0067	(7.0)	0.2018 ± 0.0041	(3.0)	0.2153 ± 0.0068	(6.0)	**0.2013** ± 0.0051	(1.0)	0.2105 ± 0.0058	(5.0)	0.2017 ± 0.0043	(2.0)	0.2162 ± 0.0067	(8.0)
Reuters	0.0458 ± 0.0008	(3.0)	0.0577 ± 0.0012	(7.0)	**0.0457** ± 0.0008	(1.5)	0.0578 ± 0.0012	(8.0)	0.0476 ± 0.0010	(4.0)	0.0478 ± 0.0011	(5.0)	**0.0457** ± 0.0008	(1.5)	0.0558 ± 0.0013	(6.0)
Scene	0.0983 ± 0.0029	(3.0)	0.1831 ± 0.0047	(8.0)	**0.0935** ± 0.0028	(1.0)	0.1699 ± 0.0048	(7.0)	0.1069 ± 0.0030	(4.0)	0.1100 ± 0.0033	(5.0)	0.0970 ± 0.0030	(2.0)	0.1600 ± 0.0044	(6.0)
Slashdot	**0.0373** ± 0.0004	(1.0)	0.0880 ± 0.0019	(8.0)	0.0374 ± 0.0005	(2.0)	0.0873 ± 0.0019	(7.0)	0.0402 ± 0.0006	(4.0)	0.0646 ± 0.0019	(5.0)	0.0379 ± 0.0005	(3.0)	0.0791 ± 0.0017	(6.0)
Yeast	0.1981 ± 0.0032	(2.5)	0.2156 ± 0.0036	(8.0)	0.1981 ± 0.0033	(2.5)	0.2149 ± 0.0037	(7.0)	0.2099 ± 0.0041	(4.0)	0.2100 ± 0.0041	(5.0)	**0.1978** ± 0.0031	(1.0)	0.2107 ± 0.0037	(6.0)
Avg. rank		(3.64)		(6.55)		(3.14)		(5.50)		(3.91)		(4.68)		(2.59)		(6.00)
Subset 0/1 *Loss*																
Bibtex	0.8283 ± 0.0043	(8.0)	0.8154 ± 0.0053	(4.0)	0.8269 ± 0.0046	(7.0)	0.8146 ± 0.0053	(2.0)	0.8211 ± 0.0049	(5.0)	**0.8141** ± 0.0057	(1.0)	0.8226 ± 0.0044	(6.0)	0.8149 ± 0.0052	(3.0)
Emotions	0.7942 ± 0.0156	(8.0)	0.7470 ± 0.0174	(5.0)	0.7825 ± 0.0158	(7.0)	0.7369 ± 0.0151	(4.0)	0.7098 ± 0.0216	(2.0)	**0.7015** ± 0.0179	(1.0)	0.7741 ± 0.0192	(6.0)	0.7319 ± 0.0164	(3.0)
Enron	0.8690 ± 0.0055	(8.0)	0.8525 ± 0.0088	(4.5)	0.8661 ± 0.0054	(7.0)	0.8525 ± 0.0088	(4.5)	0.8408 ± 0.0070	(2.0)	**0.8320** ± 0.0103	(1.0)	0.8578 ± 0.0053	(6.0)	0.8455 ± 0.0086	(3.0)
Genbase	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)	**0.0181** ± 0.0030	(4.5)
Mediamill	0.9036 ± 0.0047	(8.0)	0.8814 ± 0.0039	(6.0)	0.8794 ± 0.0051	(5.0)	0.8610 ± 0.0049	(2.0)	0.8620 ± 0.0043	(3.0)	**0.8580** ± 0.0037	(1.0)	0.8834 ± 0.0042	(7.0)	0.8624 ± 0.0039	(4.0)
Medical	0.3394 ± 0.0123	(8.0)	0.3149 ± 0.0094	(6.0)	0.3292 ± 0.0115	(7.0)	0.3067 ± 0.0091	(3.0)	0.3098 ± 0.0098	(4.0)	**0.2986** ± 0.0102	(1.0)	0.3118 ± 0.0103	(5.0)	0.3016 ± 0.0100	(2.0)
Image	0.7150 ± 0.0097	(8.0)	0.6520 ± 0.0133	(3.0)	0.7095 ± 0.0101	(6.0)	0.6500 ± 0.0139	(2.0)	0.6700 ± 0.0161	(5.0)	**0.6430** ± 0.0123	(1.0)	0.7110 ± 0.0102	(7.0)	0.6530 ± 0.0134	(4.0)
Reuters	0.2569 ± 0.0043	(8.0)	0.2460 ± 0.0042	(5.0)	0.2561 ± 0.0043	(7.0)	0.2455 ± 0.0043	(4.0)	0.2375 ± 0.0056	(2.5)	**0.2263** ± 0.0055	(1.0)	0.2484 ± 0.0049	(6.0)	0.2375 ± 0.0049	(2.5)
Scene	0.4603 ± 0.0100	(8.0)	0.4499 ± 0.0085	(7.0)	0.4275 ± 0.0098	(6.0)	0.4225 ± 0.0087	(5.0)	0.3635 ± 0.0097	(2.0)	**0.3552** ± 0.0081	(1.0)	0.4167 ± 0.0097	(4.0)	0.4072 ± 0.0093	(3.0)
Slashdot	0.6195 ± 0.0055	(6.0)	0.6285 ± 0.0063	(8.0)	0.6182 ± 0.0055	(5.0)	0.6277 ± 0.0063	(7.0)	**0.5651** ± 0.0065	(1.0)	0.5788 ± 0.0077	(2.0)	0.5949 ± 0.0056	(3.0)	0.6063 ± 0.0071	(4.0)
Yeast	0.8453 ± 0.0100	(8.0)	0.8407 ± 0.0120	(6.0)	0.8432 ± 0.0099	(7.0)	0.8378 ± 0.0118	(5.0)	**0.7865** ± 0.0119	(1.0)	0.7869 ± 0.0120	(2.0)	0.8246 ± 0.0115	(3.0)	0.8250 ± 0.0126	(4.0)
Avg. rank		(7.50)		(5.36)		(6.23)		(3.91)		(2.91)		(1.50)		(5.23)		(3.36)

Moreover, it is clear that the better such estimations are, the better the predictions of DBR will be, as we expected. But an important question remains unanswered. DBR needs another multi-label classifier method which provides label estimations. Thus, the practitioner that decides to use DBR learner must train an additional multi-label classifier. But, is it worthy? Or, stated differently, can DBR improve the performance of that multi-label classifier? When we employed BR in our first experiment, we observed that DBR(BR) outperformed BR in all measures except Hamming loss. But, what would happen with a more sophisticated multi-label approach?

In this final experiment, we used the rest of the methods studied in this paper, STA, CC and NS, to feed DBR models during the test phase. The process was simple: first, we trained once our DBR model, and then we tested it with the label estimations supplied by the other three multi-label classifiers. Recall that DBR models were exactly the same in all cases because (i) the training phase of a DBR model requires true label information, not estimations, and (ii) the parameter selection procedure (see Section 5.1) individually selects the values that optimize the accuracy for each individual binary classifier. That is, neither the training phase of DBR, nor the parameter selection process depends on the multi-label classifier used to provide label estimations, nor fit to it.

Table 5 shows the scores for F_1 and Jaccard index, while Hamming loss and Subset 0/1 loss are presented in Table 6. The goal is to prove if DBR(M) performs better than M, where M is the multi-label classifier used to provide label estimations. We have included the scores for BR and DBR(BR) as reference. Despite the average ranks are shown, we will not discuss the corresponding Nemenyi tests because there are several multi-label classifiers, all DBR versions, that have the same learning bias. This fact may alter the results of the Nemenyi test because, for instance, if DBR performs well in a particular domain, all DBR versions usually obtain a good rank, relegating the rest of the methods in that dataset. Instead, we will use the Wilcoxon sign-ranks test to compare each DBR(M) with its counterpart M.

Analyzing the results for all performance measures, we observe more or less the same conclusions that in case of DBR(BR). Each DBR version outperforms its respective counterpart method in terms of F_1, Jaccard index and Subset 0/1 loss. The number of wins, ties and losses is respectively 35/0/9, 36/0/8 and 34/4/6, for a total of 105 wins, 4 ties and 23 losses. The losses are always in the same domains: mediamill and yeast, except one loss in bibtex dataset for F_1. On the contrary, DBR versions are again worse than its counterpart learner in Hamming loss, with only 8 wins, 2 ties and 34 losses.

Table 7 summarizes the results of the Wilcoxon sign-ranks tests performed. As shown, DBR is significantly better for F_1 and Jaccard index and Subset 0/1 loss, but worse for Hamming loss. We can observe that, in all comparisons, the test rejects the null-hypothesis that both algorithms perform equally well, except in case of CC vs. DBR(CC) for Subset 0/1 loss, but the p-value obtained is quite low. In fact, the results of DBR(CC) in that particular case are pretty good: DBR(CC) improves the score of CC in 8 out of 11 datasets, when it is known that CC is well-tailored for Subset 0/1 loss, despite in some particular cases it can perform poorly [23]. Nevertheless, in our experiments, CC outperforms the rest of the methods in terms of Subset 0/1 loss, except DBR(CC) which normally obtains better scores.

Interestingly, looking at the results from another perspective, the four versions of DBR are at the top of the ranking, ahead of the rest of methods, for F_1 and Jaccard index. This does not happen for Subset 0/1 loss, because CC is the second best approach. For Hamming loss the results are just the opposite. Notice that DBR (BR), used as representative of our method in the first experiment, obtains the worse average ranks among DBR versions for all performance measures. This result is not strange because STA, CC and NS are more complex learners than BR, and it confirms our claim that the better the label estimations used, the better the performance of DBR. However, we cannot state that DBR will be able to improve the performance of any multi-label classifier used to provide the required label estimations. But, it seems that when the problem presents an important level of conditional dependence, DBR tends to obtain very good results in those measures biased to the relevant labels.

Table 7
Wilcoxon signed-ranks tests, comparing DBR with each method used to obtain labels estimations. Each row shows the p-value for each performance measure. The symbol ⇈ (⇊) means that DBR is significantly better (worse) than the other method at level $p < 0.01$ (↑ and ↓ at level $p < 0.05$).

Comparison	F_1	Jaccard Index	Hamming loss	Subset 0/1 loss
BR vs. DBR(BR)	0.0186 ↑	0.0186 ↑	0.0098 ⇊	0.0059 ⇈
STA vs. DBR(STA)	0.0420 ↑	0.0186 ↑	0.0137 ↓	0.0098 ⇈
CC vs. DBR(CC)	0.0186 ↑	0.0137 ↑	0.0186 ↓	0.0801
NS vs. DBR(NS)	0.0244 ↑	0.0244 ↑	0.0039 ⇊	0.0371 ↑

6. Conclusions and future work

Although multi-label classification can be seen as a simple extension of the well-studied single-class classification, it comes with the challenge that labels usually display dependencies amongst each other. This paper proposes the dependent binary relevance (DBR) approach to cope with multi-label classification under the hypothesis that, in general, the prediction of each label can benefit from information about the other labels. To that end, our learner employs an extended feature space composed of the original input space and the label space. The main goal of dependent binary models is to exploit conditional label dependence without making any assumption about the potential relationships among labels. Our approach requires label estimations provided by another multi-label classifier in order to make predictions. This issue has been extensively analyzed throughout the paper, concluding that the more knowledge of the presence of certain labels, the better the performance of our method.

The experiments carried out have shown that a DBR model is able to detect conditional label dependence better than other state-of-the-art multi-label classifiers, notably classifier chains (CC). This is due to the fact that DBR employs all available information. However, the method presents an important drawback, shared with CC, but even more damaging. It suffers from a problem of attribute noise during the test phase: when the label estimations used contain many mistakes, the method may perform poorly. The influence of this issue depends on the domain and, more importantly, on the evaluation measure. In our experiments, DBR performs quite well in terms of those evaluation measures that put emphasis on a correct prediction of the relevant labels. However, the attribute noise is much more harmful for Hamming loss, deteriorating the performance of our approach.

As future work, we plan to further investigate how to deal with the problem of attribute noise in a proper way, reducing its influence on the performance of dependent binary models. Maintaining the assumptions of our approach, the main idea is to somehow make the individual binary classifiers less dependent of the information from the rest of the labels, focusing mainly on the original input space, whenever object descriptions are enough to correctly predict those examples. This kind of base learners could be used not only for DBR, but also for CC and other interesting chaining approaches, including PCC.

Conflict of interest

None declared.

Acknowledgments

This research has been supported by Spanish Ministerio de Economía y Competitividad (Grant TIN2011-23558) and by the German Research Foundation (DFG).

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.patcog.2013.09.029.

References

[1] A. Clare, R.D. King, Knowledge discovery in multi-label phenotype data, in: European Conference on Data Mining and Knowledge Discovery, 2001, pp. 42–53.
[2] M.-L. Zhang, Z.-H. Zhou, Ml-knn: A lazy learning approach to multi-label learning, Pattern Recognition 40 (2007) 2038–2048.
[3] M.-L. Zhang, Z.-H. Zhou, Multilabel neural networks with applications to functional genomics and text categorization, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 1338–1351.
[4] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, in: ACM Conference on Research and Development in Information Retrieval, 2005, pp. 274–281.
[5] A.K. McCallum, Multi-label text classification with a mixture model trained by EM, in: AAAI 99 Workshop on Text Learning.
[6] N. Ghamrawi, A. McCallum, Collective multi-label classification, in: ACM International Conference on Information and Knowledge Management, ACM, 2005, pp. 195–200.
[7] R.E. Schapire, Y. Singer, Boostexter: a boosting-based system for text categorization, Machine Learning (2000) 135–168.
[8] K. Dembczyński, W. Waegeman, W. Cheng, E. Hüllermeier, On label dependence and loss minimization in multi-label classification, Machine Learning 88 (2012) 5–45.
[9] K. Dembczyński, W. Cheng, E. Hüllermeier, Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains, in: ICML, 2010, pp. 279–286.
[10] E. Montañés, J.R. Quevedo, J.J. del Coz, Aggregating independent and dependent models to learn multi-label classifiers, in: ECML/PKDD'11—Volume Part II, Springer-Verlag, 2011, pp. 484–500.
[11] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Machine Learning 85 (2011) 333–359.
[12] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining multi-label data, in: Data Mining and Knowledge Discovery Handbook, 2010, pp. 667–685.
[13] W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Machine Learning 76 (2009) 211–225.
[14] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004, pp. 22–30.
[15] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, K. Brinker, Multilabel classification via calibrated label ranking, Machine Learning 73 (2008) 133–153.
[16] G.J. Qi, X.S. Hua, Y. Rui, J. Tang, T. Mei, H.J. Zhang, Correlative multi-label video annotation, in: Proceedings of the International Conference on Multimedia, ACM, NY, USA, 2007, pp. 17–26.
[17] J. Read, B. Pfahringer, G. Holmes, Multi-label classification using ensembles of pruned sets, in: IEEE International Conference on Data Mining, IEEE, 2008, pp. 995–1000.
[18] G. Tsoumakas, I. Vlahavas, Random k-labelsets: an ensemble method for multilabel classification, in: ECML/PKDD'07, Lecture Notes in Computer Sciences, Springer, 2007, pp. 406–417.
[19] E. Montañés, J.R. Quevedo, J.J. del Coz, Improving stacking approach for multi-label classification, in: Proceedings of the 2011 Spanish Conference on Artificial Intelligence, pp. 484–500.
[20] Ó. Luaces, J. Díez, J. Barranquero, J.J. del Coz, A. Bahamonde, Binary relevance efficacy for multilabel classification, Progress in Artificial Intelligence 4 (2012) 303–313.
[21] K. Dembczyński, W. Waegeman, W. Cheng, E. Hüllermeier, Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss, in: ECML/PKDD'10, Part I, Springer, 2010, pp. 280–295.
[22] R. Senge, J.J. del Coz, E. Hüllermeier, On the problem of error propagation in classifier chains for multi-label classification, in: Conference of the German Classification Society on Data Analysis, Machine Learning and Knowledge Discovery, 2012.
[23] K. Dembczyński, W. Waegeman, E. Hüllermeier, An analysis of chaining in multi-label classification, in: ECAI 2012, pp. 294–299.
[24] A. Kumar, S. Vembu, A.K. Menon, C. Elkan, Learning and inference in probabilistic classifier chains with beam search, in: ECML/PKDD 2012, pp. 665–680.
[25] D.H. Wolpert, Stacked generalization, Neural Networks 5 (1992) 214–259.
[26] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, I. Vlahavas, Correlation-based pruning of stacked binary relevance models for multi-label learning, in: Workshop on Learning from Multi-Label Data, 2009, pp. 101–116.
[27] R. Senge, J.J. del Coz, E. Hüllermeier, Rectifying classifier chains for multi-label classification, in: A. Henrich, H.-C. Sperker (Eds.), Lernen, Wissen, Adaption (LWA) 2013, Proceedings, 2013, pp. 162–169.
[28] X. Zhu, X. Wu, Class noise vs. attribute noise: a quantitative study of their impacts, Artificial Intelligence Review 22 (2004) 177–210.
[29] T.G. Dietterich, Ensemble methods in machine learning, in: Proceedings of the First International Workshop on Multiple Classifier Systems, Springer-Verlag, London, UK, 2000, pp. 1–15.
[30] W.W. Cohen, V.R. de Carvalho, Stacked sequential learning, in: International Joint Conference on Artificial Intelligence (IJCAI), 2006, pp. 671–676.
[31] G. Madjarov, D. Kocev, D. Gjorgjevikj, S. Džeroski, An extensive experimental comparison of methods for multi-label learning, Pattern Recognition 45 (2012) 3084–3104.
[32] C.-J. Lin, R.C. Weng, S.S. Keerthi, Trust region Newton method for logistic regression, Journal of Machine Learning Research 9 (2008) 627–650.
[33] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

Elena Montañes received her M.Sc. degree in Mathematics in 1998 and her Ph.D. in Computer Science from the University of Oviedo, Spain, in 2003. She is currently a senior Lecturer and a member of the Artificial Intelligence Center, Gijón. Her research interests are focused on several machine learning problems, e.g. multi-label and multi-instance tasks.

Robin Senge received his M.Sc. degree in Computer Science from the University of Marburg, Germany, in 2006. After graduation, he worked as a software engineer in industry, where he was consigned to consulting and developed software for financial applications. In 2009, he joined the Computational Intelligence Group (CIG) at the University of Marburg as a doctoral student. His research interests are focused on machine learning and fuzzy systems.

Jose Barranquero is a predoctoral researcher at Artificial Intelligence Center (University of Oviedo). He has a M.Sc. in Web Engineering and a M.Sc. in Soft Computing and Intelligent Data Analysis from the University of Oviedo. His research interests include machine learning, quantification, opinion mining and microblogging networks.

José Ramón Quevedo received his M.Sc. and his Ph.D. degrees in Computer Science from the University of Oviedo, Spain, in 1997 and 2000 respectively. He is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. His current research is focused on applying machine learning methods to bioinformatics applications.

Juan José del Coz received his Ph.D. degree in Computer Science from the University of Oviedo at Gijón, Spain, in 2000. In 1997, he joined the Computer Science Department of the University of Oviedo, where he is currently a Tenured Associate Professor. He has authored over thirty papers in peer reviewed journals and conferences including articles in NIPS, ICML, JMLR and Pattern Recognition.

Eyke Hüllermeier is with the Department of Mathematics and Computer Science at Marburg University (Germany), where he holds an appointment as a full professor and heads the Computational Intelligence Group. In research, he is mostly interested in machine learning, uncertainty in knowledge-based systems, and applications in bioinformatics.