Noname manuscript No.
(will be inserted by the editor)

Learning Rules for Multi-label Classification: a Stacking and a Separate-and-Conquer Approach

Eneldo Loza Mencía · Frederik Janssen

the date of receipt and acceptance should be inserted later

Abstract Dependencies between the labels are commonly regarded as the crucial issue in multi-label classification. Rules provide a natural way for symbolically describing such relationships. For instance, rules with label tests in the body allow for representing directed dependencies like implications, subsumptions, or exclusions. Moreover, rules naturally allow to jointly capture both local and global label dependencies. In this paper, we introduce two approaches for learning such label-dependent rules. Our first solution is a bootstrapped stacking approach which can be built on top of a conventional rule learning algorithm. For this, we learn for each label a separate ruleset, but we include the remaining labels as additional attributes in the training instances. The second approach goes one step further by adapting the commonly used separate-and-conquer algorithm for learning multi-label rules. The main idea is to re-include the covered examples with the predicted labels so that this information can be used for learning subsequent rules. Both approaches allow for making label dependencies explicit in the rules. In addition, the usage of standard rule learning techniques targeted at producing accurate predictions ensures that the found rules are useful for actual classification. Our experiments show a) that the discovered dependencies contribute to the understanding and improve the analysis of multi-label datasets, and b) that the found multi-label rules are crucial for the predictive performance as our proposed approaches beat the baseline using conventional rules.

Keywords Multi-label Classification, Rule Learning, Stacking, Label Dependencies

Technische Universität Darmstadt
Knowledge Engineering Group
E-mail: {eneldo,janssen}@ke.tu-darmstadt.de

1 Introduction

Rule learning has a very long history and is a well-known problem in the machine learning community. Over the years many different algorithms to learn a set of rules were introduced. The main advantage of rule-based classifiers are interpretable models as rules can often be easily comprehended by humans. Also, rules form a structured hypothesis space which allows to easily generalize or specialize individual rules. Thus, rule models can be simply modified and processed as opposed to most statistical models such as SVMs or neural networks.

Many problems involve assigning more than a single class to an object. Such so-called multi-label problems can often be found when text is assigned with different topics or tagged with keywords, but there are also many examples from other domains such as the identification of music instruments or emotions in audio recordings or the classification of scenes in images [46].

It is widely accepted that one major issue in learning from such multi-label data is the exploitation of label dependencies [53, 19]. Learning algorithms may greatly benefit from considering label correlations, and we believe that rule induction algorithms are best suited for exploiting these in a simple way. Firstly, so called global dependencies between only labels can be explicitly modeled and expressed in the form of rules. Moreover, and possibly more interestingly, dependencies that include both label and regular features can be represented, which we refer to as local dependencies [12]. Secondly, such rules are directly interpretable and comprehensible for humans. Even if complex and long rules are generated, the implication between labels can be grasped more easily than with other approaches by focusing on the part of the rules that actually considers the labels. Hence, in contrast to many other model types, which capture class dependencies implicitly, an explicit notation allows to analyze and interpret them directly. Thirdly, the usage of rule induction algorithms particularly targeted at classification ensure that effective predictive models are produced rather than only descriptive models.

We introduce a formalization of the different types of multi-label rules in which we differentiate between the presence of label assignments and label tests in the heads and bodies of the different types of multi-label rules. The type of rule a learner can induce also determines the type of label dependencies which be can discovered and hence exploited by the used method. In this work, we focus on learning so called label-dependent single-label rules, which assign values only to a single label but can include conditions on other labels in the body. We argue that this simple type of rule allows for adequately expressing many different types of label dependencies such as implications, subsumptions and exclusions.

We propose two different approaches in this work in order to learn such rules. In the first one the true label information is directly provided to the rule learner. This is done by stacking the label features as additional input instance features [30]. The result is an independent ruleset for each label. Although the proposed stacking method is not the first approach in making use of stacking in order to consider label dependencies (cf. Section 3), it is to our knowledge the first time rule induction was used in order to make label dependencies explicit. We show that stacking, though conceptually very simple, is suitable in order to reveal global as well as local label dependencies. Moreover, the induced models allow for a detailed analysis of the datasets w.r.t. the contained dependencies. For instance, statistics on the induced rulesets allow to (empirically) estimate the degree of dependencies.

This may lead to a better understanding of the dataset at hand. We provide an extensive analysis for eleven datasets commonly used by the community for benchmarking.

Our second, much more versatile approach, is to integrate the label information directly into a joint rule learning process. More precisely, we adapt the separate-and-conquer learning procedure in an iterative and recursive manner: Whenever the algorithm has learned a new rule for predicting a particular label, this rule is directly applied to the set of training instances and therefore the prediction of the rule becomes available for the following covering iterations. The result is a global joint ruleset for all labels in which all necessary dependencies are integrated as they are needed for the classification.

The resulting model is usually a much more compact representation of the dataset than the one produced by our first approach. On the other hand, our second proposition is less effective in revealing label dependencies since the necessary information only becomes available piece by piece in the course of the learning process.

Although we are interested in comprehensive and descriptive models for a multi-label classification task at hand, an important requirement for such models is still that they are accurate and hence effective for classification. Consequently, both approaches primarily aim at producing high-quality classification models. The objective of obtaining expressive rules is achieved only by extending the hypothesis space to multi-label rules. In the experimental evaluation, we show that the predictive performance does indeed not suffer from the additional objective. On the contrary, the stacking approach is able to improve over the baseline rule learner which is not able to induce multi-label rules.

In the following Section 2 we introduce multi-label classification and inductive rule learning and provide a formal framework for their integration. Additionally, we give a formal definition of different types of multi-label rules. Then, we explain how stacking label features can be used in order to learn multi-label rules and how these rules can be used for prediction (cf. Section 3). We introduce *multi-label iterative separate-and-conquer* and *multi-label decision lists* in detail in Section 4. We then discuss the most relevant related work in Section 5 before we continue with the evaluation in Section 6 that focuses on inspecting and analyzing the rule models, but also provides an extensive comparison of both approaches. First experiments aiming to compare the two approaches to other rule-based methods for multi-label classification are also shown. In Section 7 we conclude with a summarization and directions for further work.

2 Multi-Label Classification and Inductive Rule Learning

For the sake of a successful combination of multi-label classification and inductive rule learning, we briefly introduce both disciplines in this section. Starting with a description of and a motivation for multi-label learning, we then relate our approach to existing research. As a key advantage of multi-label rule learning is the ability to find conditional dependencies, such local dependencies are introduced via an example and their difference to global dependencies is sketched. Then, inductive rule learning is introduced with a special focus on the different types of rules that are necessary to generalize them for multi-label learning.

2.1 Multi-label Classification

Multi-label classification (MLC) refers to the task of learning a function $h(\mathbf{x})$ that maps instances $\mathbf{x} = (x_1, \ldots, x_a) \in \mathcal{X}$ to label subsets or label vectors $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$, where $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$, $n = |\mathcal{L}|$ is a finite set of predefined labels and where each label attribute y_i corresponds to the absence (0) or presence (1) of label λ_i . Thus, in contrast to multiclass learning, alternatives are not assumed to be mutually exclusive, such that multiple labels may be associated with a single instance. This, and especially the resulting correlations and dependencies between the labels in $\mathcal{L}$, make the multi-label setting particularly challenging and interesting compared to the classical field of binary and multiclass classification.

From a probabilistic point of view, this is one of the main differences. In binary and multiclass problems the only observable probabilistic dependence is between the input variables, i.e., the attributes x_j, and the label variables y_i. A learning algorithm tries to learn exactly this dependence in form of a classifier function h. In fact, if a classifier provides a score or confidence for its prediction $\hat{\mathbf{y}} = h(\mathbf{x})$, this is often regarded as an approximation of $P(\mathbf{y} = \hat{\mathbf{y}} \,\big|\, \mathbf{x})$, i.e., the probability that $\hat{\mathbf{y}}$ is true given a document $\mathbf{x}$.

The predominant approach in multi-label classification is *binary relevance* (BR) learning [cf., e.g., 48]. It tackles a multi-label problem by learning one classifier for each label, using all objects of this label as positive examples and all other objects as negative examples. There exists hence a strong connection to concept learning, which is dedicated to infer a model or description of a target concept from specific examples of it [see, e.g., 10]. When several target concepts are possible or given for the same set of instances, we formally have a multi-label problem. The problem of this approach is that each label is considered independently of each other, and as we have seen by the example given before, this can lead to loss of useful information for classification.

2.1.1 Label Dependencies

Therefore, from the early beginning of multi-label classification, there have been attempts to exploit these types of *label correlations* [e.g. 34, 18, 54]. A middle way is followed by Read et al. [40] and Dembczyński et al. [12] in their popular (probabilistic) classifier chains by stacking the underlying binary relevance classifiers with the predictions of the previous ones. However, only recently Dembczyński et al. provided a clarification and formalization of label dependence in multi-label classifications. Following their argumentation, one must distinguish between unconditional and conditional label dependence. Roughly speaking, the *unconditional dependence* or independence between labels does not depend on a specific given input instance (the condition) while *conditional dependence* does. We may also refer to these as *global* and *local* dependencies, since they are revealed globally or only in subspaces of the input space.

An example may illustrate this: Suppose a label space indicating topics from news articles and a subtopic *foreign affairs* of the topic *politics*. Obviously, there will be a dependency between both labels, since the presence of a subtopic implies the presence of the super topic and the probability of *foreign affairs* would be higher than average if *politics* is observed. These probabilities are *unconditional* or *global* since they do not depend on a particular document. Suppose now that

a particular news article is about the *Euro crisis*. Under this condition, the *conditional* probabilities for both labels as well as the dependency between would likely increase and hence be different from the unconditional ones. However, if an article was about the *cardiovascular problems of Ötzi*, we would observe that both labels are *conditionally independent* for this instance, since the probability for one label would very likely not depend on the presence of the other label (both being very low).

2.2 Inductive Rule Learning

As the goal of this paper is to make label dependencies explicit by using rules, we will also shortly introduce inductive rule learning. This is one of the oldest and therefore best researched fields in machine learning. A rule learning algorithm is used to learn a set of classification rules $\mathcal{R}$. These rules r are of the form

$$head \leftarrow body$$

where the body consists of a number of *conditions* (attribute-value tests) and, in the regular case, the head has only one single condition of the form $y_i = 0$ or 1 (in our case). Similarly to the popular *Ripper* rule learner [9], which is still the state-of-the-art for rule induction, we consider only conjunctive rules in this work.

Ripper uses the *separate-and-conquer* (SeCo) or *covering* approach [15] for inducing a set of rules for *concept learning* problems, where instances belonging to a certain concepts have to be differentiated against those that are not. Such separate-and-conquer algorithms proceed by firstly learning a single rule on the data. Once a rule is found, it is added to the ruleset and all examples covered by this rule are removed. Then, the next rule is learned and the algorithm keeps inducing new rules as long as (positive) examples are left in the dataset. How a single rule is learned is described in detail in Section 4.4 and in Algorithm 4. In order to prevent overfitting, the two constraints that all examples have to be covered (*completeness*) and that negative examples must not be covered (*consistency*) can be relaxed so that some positive examples may remain uncovered and/or some negative examples may be covered by the set of rules. In the following, we use p and n to denote the covered positive and negative examples whereas P and N stand for the total number of positive and negative examples. The SeCo strategy only works for binary datasets. Hence, a natural way of addressing multi-label problems is to consider each label separately (cf. Section 2.1), resulting in a model consisting of separate rulesets for each label.

A SeCo algorithm either returns an ordered or an unordered list of rules. The former is a *decision list* where the rules are checked in order and whenever the rule at hand covers an example it is used to predict the class of the example. All subsequent rules are skipped for this example. In theses cases, we assume an ordering on the rules in the ruleset and overload the notation so that $\mathcal{R} = \langle r_1, r_2, \ldots \rangle$.

Some algorithms process the dataset by ordering the classes by their frequency and then start learning rules for the smallest class. In the end, rules for the largest class are not learned and a *default rule* is added to the bottom of the list which simply assigns all instances to the largest class. Then, for binary datasets, this means that such a rule learner learns a concept only for the *smaller* class.

In Section 4, we additionally introduce and make use of *multi-label decision lists*, which are able to express MLC rule models in a single, joint decision list.

2.3 Different Types of Multi-label Rules

Table 1 gives an overview of the types of multi-label rules used in the remainder. As mentioned above, we use conjunctive rules in this work. For each rule type, example rules are shown which may contain hypothetical conditions c_1, c_2, and c_3 and labels y_1, y_2 and y_3. We concentrate on learning *single-label head* rules where only one label appears in the head of the rule in contrast to *multi-label head* rules (last two rows in Table 1), which contain several label assignments in their head. *Multi-label head* rules conveniently allow to model co-occurrence dependencies between labels. Often, it is beneficial to omit the prediction for some of the labels instead of predicting a full label combination, e.g., for postponing the prediction for subsequent rules. In that case, we refer to them as *sparse* multi-label head rules instead of *dense* ones.

Commonly, the conditions in the body are tests on attributes from the instance space (cf. *label-independent* rules with conditions c_i in Table 1). However, in order to reflect label dependencies (e.g., implications, subsumptions, or exclusions), we would need to have labels on both sides of the rule. Hence, if a rule may contain conditions on the labels, we refer to it as a label-dependent rule (also called *contextual* rule by D. Malerba and Esposito [10]), and *label-independent* if this is not the case. Global dependencies are hence best reflected by *fully label-dependent* bodies, whereas local dependencies can be described by *partially label-dependent* rules with regular conditions and label tests in the body (cf. Table 1).

Another distinction of rules comes from whether the presence ($y_i = 1$) or the absence of a label ($y_i = 0$) is used for prediction or in the body of the rule (in Table 1 denoted by y_i and $\overline{y}_i$, respectively). Since multi-label datasets are usually very asymmetric in the sense that labels appear relatively infrequently (see Section 4.4 for a discussion), it is common in MLC to focus on detecting the presence of labels. Also, from the perspective of concept learning (cf. Section 2.1), it is reasonable to learn a description of a label (in form of rules) for the case that the label is present. Hence, multi-label rule learning approaches are often restricted to learning only *positive head* rather than *positive and negative head* rules. However, in some situations, it may be beneficial to leave it open whether the rule predicts the

Table 1 Different forms of multi-label rules. Let the number of labels be $n = 3$, y_i denote positive label assignments or positive label tests ($y_i = 1$), over-lined $\overline{y}_i$ denote negative assignments or tests ($y_i = 0$) , respectively, and let c_1, c_2, and c_3 be some conditions on instance features x_i.

head		body	example rule
single-label	positive negative	label-independent	$y_1 \leftarrow c_1, c_2, c_3$ $\overline{y}_1 \leftarrow \overline{c}_1, c_2$
single-label	positive negative	partially label-dependent	$y_3 \leftarrow c_1, \overline{y}_1, y_2$ $\overline{y}_3 \leftarrow \overline{c}_1, \overline{y}_1, y_2$
single-label	positive negative	fully label-dependent	$y_3 \leftarrow \overline{y}_1, y_2$ $\overline{y}_3 \leftarrow y_1, \overline{y}_2$
multi-label	sparse dense	label-independent	$y_1, y_2 \leftarrow c_1, c_2, c_3$ $y_1, \overline{y}_2, \overline{y}_3 \leftarrow c_1, c_2, c_3$

Require: New training example pairs $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$,
targets $\mathcal{B}$ (either $\mathcal{B} = \{1\}$ or $\mathcal{B} = \{0, 1\}$)

for each label y_i **do**
  $\mathcal{T}_i \leftarrow \emptyset$
  for each $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}$, $\mathbf{x} = (x_1, \ldots, x_a)$, $\mathbf{y} = (y_1, \ldots, y_n)$ **do**
    $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup ((x_1, \ldots, x_a, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n), (y_i))$
  $\mathcal{R}_i^* \leftarrow findRules(y_i, \mathcal{B}, \mathcal{T})$ ▷ learn rule list for label y_i
return $\mathcal{R}_i^*$ ▷ stacked rules

Fig. 1 Stacking algorithm for obtaining label-dependent single-label rules.

presence or the absence of a label. Note that in our approach either a positive label can be predicted or it is dynamically decided whether the label is predicted positive or negative.

Both, the extension of the head from single-label to multi-label assignments, and the consideration of labels in the body, can facilitate rules and rule models to represent dependencies between labels. However, there are two major reasons why we focus on the induction of label-dependent rules. Firstly, from a conceptual point of view, multi-label head rules can only comprehensively model co-occurrence (or co-absence) dependency cases, whereas we are interested in a broader range of dependency types. A consequence of this limitation is also the difficulty in expressing global dependencies. Consider, e.g., the simple dependency $y_2 \leftarrow y_1$ often appearing in hierarchical structures. It is not possible to adequately express this global relationship in form of a simple rule. Instead, a learner would have to describe both cases $y_2 = 1, y_1 = 0$ and $y_2 = 1, y_1 = 1$ separately by different rulesets, concealing the obvious connection between both labels and the global character of it. The situation is similar for expressing local dependencies which are not co-occurrences. For instance, representing the dependency $y_2 \leftarrow y_1, c_1$ as multi-label head rule with label-independent bodies would need to add all rules for y_1 again to the ruleset with c_1 as additional condition.

The reason for the increased complexity is connected to a further disadvantage, namely that representing a multi-label dataset with multi-label rules may require to formulate separate rules for each distinct label combination present in the dataset. The maximum number of distinct combinations grows exponentially with the number of labels or linearly with the number of training examples ($\mathcal{O}(min(2^n, m))$, Hence, the number of necessary rules can become very large for datasets of medium size. We believe that single-label head rules, in contrast, can be used to obtain more compact classification and descriptive rule models since labels can be described independently from each other as well as by using the shared descriptions of other labels, as needed. The full expressiveness is though obtained by *label-dependent multi-label* rules, which we leave for further research.

In summary: We start from *label-independent single-label* rules and move towards *label-dependent single-label* rules, which, as shown, are well suited for modeling and expressing label dependencies. We present in the next sections two approaches particularly suited in order to find such rules. Both approaches are able to find *positive* as well as *positive or negative single-label* rules.

Require: Text example $\mathbf{x}$,
decision lists $\mathcal{R}_i^*$, decision functions h_i^* based on $\mathcal{R}_i^*$,
number of bootstrapping iterations m,
initialization $\hat{\mathbf{y}}_0 = (\hat{y}_{0,1}, \hat{y}_{0,2}, \ldots, \hat{y}_{0,n})$ (e.g., predictions from single-label models $\mathcal{R}_i$)
1: **for** j in $1 \ldots m$ **do**
2: **for each** label y_i **do**
3: $\hat{y}_{j,i} \leftarrow h_i^*(x_1, \ldots, x_a, \hat{y}_{j-1,1}, \ldots, \hat{y}_{j-1,i-1}, \hat{y}_{j-1,i+1}, \ldots, \hat{y}_{j-1,n})$ ▷ apply $\mathcal{R}_i^*$
4: **return** prediction $\hat{\mathbf{y}}_j = (\hat{y}_{j,1}, \hat{y}_{j,2}, \ldots, \hat{y}_{j,n})$

Fig. 2 Stacking algorithm for obtaining label-dependent single-label rules.

3 Stacking of Label Features for Learning Label-Dependent Rules

The recently very popular classifier chains [40] were found to be an effective approach for exploiting conditional label dependencies. Classifier chains (CC) make use of stacking the previous BR classifiers' predictions in order to implement the chain rule $P(y_1, \ldots, y_n) = P(y_n \mid y_1, \ldots, y_{n-1})$ in probability theory, since they learn the binary classifiers h_i with training examples of the form $(x_1, \ldots, y_1, \ldots, y_{i-1})$ [cf. 12]. One drawback of CC is the (randomly chosen) predetermined, fixed order of the classifiers (and hence the labels) in the chain, which makes it impossible to learn dependencies in the contrary direction. This was already recognized by D. Malerba and Esposito [10] in 1997, who built up a very similar system in order to learn multiple dependent concepts. In this case, the chain on the labels was determined beforehand by a statistical analysis of the label dependencies. Still, using a rule learner for solving the resulting binary problems would only allow to induce rules between two labels in one direction. Real world datasets do not have to comply with this restriction and hence we believe that for many datasets there is no possible sequence which can capture all present dependencies.

3.1 Stacked Binary Relevance

Thus, we propose to use a full stacking approach in order to overcome the main disadvantage of CC, i.e., the fixed order [30]. Such as in binary relevance, we learn one theory for each label, but we expand our training instances by the label information of the other labels, i.e., the training examples vectors for learning label y_i are of the type $(x_1, \ldots, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)$ for an instance $\mathbf{x}$. Hence, we may obtain theories with label attributes in the body, as it would be in CC. The result of using this as training data is exactly what we are seeking for, namely label-dependent single-label rules. The amount of label-features in the body additionally allows us to determine the type of dependency. We refer to this technique as *stacked binary relevance* (SBR) in contrast to plain, unstacked BR. The learning algorithm for obtaining the stacked models $\mathcal{R}_i^*$ is visualized in Figure 1.

As already mentioned, rule learners commonly learn only rules for the smaller class. SBR would hence commonly only produce positive head rules. In addition to our previous work [30], we also evaluate a variant which induces positive and negative head rules ($\mathcal{B} = \{0, 1\}$).

Stacked binary relevance is very similar to the approaches of Godbole and Sarawagi [20], Tsoumakas et al. [47], Guo and Gu [21], and very recently, Montañés

et al. [35]. They all have in common that they are using label presence information (either directly from the training data, or from the outputs of underlying BR classifiers) as (either sole or additional) features in order to learn an ensemble of binary relevance classifiers on top. The closest related approaches to our proposition are the conditionally dependency networks (CDN) [21] and the dependent binary relevance (DBR) models [35]. Both learn their models as indicated before but with one major difference: Since they are concerned with estimating probability distributions (especially joint distribution), they both use logistic regression as their base classifier, which is particularly adequate for estimating probabilities. This type of models are obviously much harder to comprehend than rules, especially for higher number of input features. Therefore, the label dependencies would remain hidden somewhere in the model, even though they may have been taken into account and accurate classifiers may have been obtained. To make the dependencies explicit and at the same time keep a high prediction quality, we propose to use rule-based models.

One additional difference between the approaches is how the prediction is conducted, which is discussed next.

3.2 Classification by Bootstrapping

For the prediction we propose to use a bootstrapping approach in the sense that we apply our models iteratively on our own previous predictions until the predictions are stable or any other stopping criterium is met. More formally, we use the learned models $\mathcal{R}_i^*$ and their associated decision functions h_i^* to produce predictions $\hat{\mathbf{y}}_j = (\hat{y}_{j,1}, \hat{y}_{j,2}, \ldots)$ where $\hat{y}_{j,i} = h_i^*(\mathbf{x}, \hat{\mathbf{y}}_{j-1})$ is based on the predictions of the previous iteration $j-1$. The pseudocode of the approach in given is Figure 2.

One obvious issue with this approach is the initialization of $\hat{\mathbf{y}}_0$. A possible option, also proposed by DBR, is to use the predictions of a BR ensemble, i.e., $\hat{y}_{0,i} = h_i(\mathbf{x})$, which is the option we mainly use in this work. The BR rulesets $\mathcal{R}_i$ are learned similarly to $\mathcal{R}_i^*$ but without the expansion by label features (using only $((x_1, \ldots, x_a), (y_i))$ in line 4 of Figure 1). Hence, each label will be characterized by two sets of rule models, namely the rules $\mathcal{R}_i$ which depend only on instance features, and a second set of rule models $\mathcal{R}_i^*$ depending (possibly) also on other labels, for which the rule models of the other labels provide predictions for.

It may happen during the classification process that no rule fired for a particular label and test instance. In that case, common rule learners would predict the majority class (for multi-label datasets, hence, typically the absence of the label). This may harm the stacking phase, since it may considerably influence the prediction of the other labels even though the evidence of the decision was rather low. Thus, we make use of the capability of abstaining from classifying if no appropriate rule was found (instead of applying the default rule). The label attribute may be then filled up in consequent iterations when more evidence exist for the decision.

3.2.1 Gibbs Sampling

In previous work, we also evaluated the option of initializing with *unknown* label information, i.e., $\hat{\mathbf{y}}_0 = (?, ?, \ldots)$, and to benefit from the natural support of sym-

bolic approaches for such attribute states (*missing*, *don't care*, etc.). On the other hand, this approach only works if the rule learner found enough rulesets with label-independent rules so that the bootstrapping can proceed and the label attributes can be filled up subsequently. In fact, this approach severely faced cold-start and deadlock problems, especially the more label-dependent rules were found by the stacking approach.

An alternative to empty initialization is to randomly initialize the label features for the bootstrapping phase, as proposed by Guo and Gu [21] and their CDN. For each random initialization, the bootstrapping approach is applied a certain number of times or until convergence (*burn-in* phase). The resulting burnt-in prediction vectors are then averaged (for binary targets voting can be used). The expectation is that random initializations closer to the true label vector tend to be corrected and converge towards the true labels vector while initializations more distant result in noise, which is eventually dominated by the voting mass of the more accurate prediction samples. Together with enough iterations (e.g., 1000) of this *Gibbs sampling*, this approach was shown to be very effective when being used with linear classifiers. We expect, however, a reduced performance for symbolic approaches due to the higher sensibility to (massive) feature noise, given in this case by the random initialization of the label features.

We had to slightly modify the original approach not only in order to support discrete binary predictions, but also solely binary inputs for the label features. In contrast to CDN which use uniform sampling out of $[0, 1] \subset \mathbb{R}$, we randomly initialize with 0 or 1 according to the prior label distribution.

3.3 Discussion and Limitations

Stacked binary relevance uses the real label information for training, although during prediction we use the predictions from the base BR classifiers. This violates to a certain degree the principle that the training data should be representative for the test data, which is certainly not the case since input features $\mathbf{y}$ and $\hat{\mathbf{y}}$ are drawn from different (but not independent) populations.

Hence, in classical stacking, the high-level classifier usually does not use the true information $\mathbf{y}$, but predictions $\hat{\mathbf{y}}'$ obtained by applying the base ensemble learners on the training data. Depending on the application and base learners used, the predictions $\hat{\mathbf{y}}'$ are obtained through a hold-out cross validation in order to get more accurate estimations of $\hat{\mathbf{y}}$ (out-of-sample in contrast to within-sample training). This is necessary since the main goal of a traditional stacking classifier is to correct and combine the predictions of the underlying classifiers in order to produce more accurate joint predictions. Certainly, the SBR learner could be applied in such a way.

However, one of the reasons for stacking the label information in the proposed algorithm is our interest in discovering and exploiting the true relationships and interconnections between labels. We are particularly interested in modeling dependencies $y_i \leftarrow y_j, \ldots$ and not $y_i \leftarrow \hat{y}_j, \ldots$. Hence, in contrast to the general understanding in traditional stacking, we do not try to estimate the features $\hat{\mathbf{y}}'$ of the training instances by the real $\mathbf{y}$, but actually we try to estimate the real $\mathbf{y}$ on the test instances by the predictions $\hat{\mathbf{y}}'$ (a different option is, e.g., to randomly guess, such as for Gibbs sampling). By doing so, the feature noise of the label

features, introduced by the estimations of the base classifiers, is transferred from training to testing time. Hence, we also expect to obtain models which reflect the relationships in a dataset more accurately and more compact than if trained on noisy predicted features.

Senge et al. [42] analyzed the effect of error propagation for classifier chains. They concluded that the problem increases the more labels are involved and lower the (labe-wise) prediction accuracy. Hence, they proposed to use (within-sample) predictions for training. On the other hand, the same authors also show that error propagation is not that severe for the dependent binary relevance algorithm, which essentially corresponds to our stacked approach, since errors can only be reinforced once (or number of bootstrapping iterations m in our case) instead of n times as for CC [35]. In their extensive analysis, it was also observed that using the real label information outperforms stacking the predictions especially for measures such as F1 which are concerned with the correct prediction of relevant labels. It became apparent that the positive effect of using the full, true and noiseless label information and hence producing better models outbalances the harmful effect of feature noise introduced by inaccurate base classifiers. This advantage is especially clearly visible for the dependent binary relevance approach whose classifiers have the full label information available rather than being restricted towards one direction of the chain. On the other hand, in their experiments the feature noise was especially harmful for measures which do not distinct between relevant and irrelevant labels, such as Hamming loss. For this loss, the approaches using traditional stacking dominated their counterparts using the true label information. Note that logistic regression was used as base classifier in both studies.

A limitation of the stacking approach may appear when circular dependencies exist between labels. A very simple example between only two labels is the relationship $\overline{y_i} \leftarrow y_j, \overline{y_j} \leftarrow y_i$, i.e., y_i and y_j excluding each other. We can assume that the rule learner would perfectly induce these rules, which is what is desired from the perspective of knowledge discovery. However, these rules do not provide any additional value for actual classification since they would just *copy* the predictions of other labels. In addition, such rules may cause endless circles or deadlocks. For instance, if $\hat{y}_{0,i}$ and $\hat{y}_{0,j}$ are initialized to $\hat{y}_{0,i} = \hat{y}_{0,j}$, either by contradicting BR classifiers or by random (Gibbs sampling), the predictions for y_i and y_j will flip in each round of the bootstrapping. [1]

Although we did not observe any harmful case of circular dependencies in our classification experiments, it may still be desirable to obtain more coherent and consistent models specifically for classification. Using classifier chains could solve the problem for circular dependencies by imposing an ordering on the predictions but with the already discussed disadvantages for detecting and exploiting the remaining (non-circular) dependencies. The separate-and-conquer approach proposed in the next section solves the issue by inducing an ordering on the predictions of labels depending on the data at hand. Hence, whether making a prediction first for either $y_{0,i}$ or $y_{0,j}$ depends on the data set (the ordering in the induced rule list) and on the instance to be classified (the covering rules).

[1] We chose a very simple global pairwise relationship for demonstrating the possible issues. The same or similar effects could appear if the relationship would only exist in subspaces of the input space (conditional dependencies), for relationships between more than two label or more complex relationships such as hierarchical relationships.

Independently of the issues originating from stacking and label dependencies, the proposed approach inherits the limitations of the base rule induction algorithm used. For example, we experienced scalability issues regarding an increasing number of instance features when using Ripper (Sec. 4.4) as rule learner. On the other hand, the additional label features did not affect the computational costs in the same manner due to their binary and sparse nature.

4 Iterative Separate-and-Conquer for Learning Multi-label Decision Lists

The presented approaches for learning multi-label models, binary relevance and the stacked variant, and also other approaches such as classifier chains, have one aspect in common, namely that they transform the original problem into several subproblems, which are then solved independently. This might be appropriate or even advantageous for certain use cases, for instance, when the objective is to obtain isolated theories representing each label (cf. concept learning), or when we try to achieve a high efficiency. However, often it is more desirable to obtain one global theory comprehensively explaining a particular multi-label dataset. The induction of one global model may also allow a better control over the objective loss, which is an important issue in multi-label classification due to the variety of existing measures, resulting directly from the diversity of the real life scenarios. In Section 4.1, we propose to use a single *multi-label decision list* as a global model.

Regarding the introduced stacked BR approach which we used for learning label-dependent single-label rules, we propose to integrate the stacking of label features directly into the SeCo rule induction process (Section 4.2). The idea is to start with unknown label features, thus, only label-independent rules will be learned in the beginning as unknown features are treated as never covered. However, the covered instances are not separated, but instead labeled accordingly and re-added to the training set.

During prediction, the rules in the multi-label decision list are applied until the first rule fires. But instead of continuing with the next test instance as for traditional decision lists, the assignments in the head are executed (for each rule a label is set in the test instances), and the process continues with the next rule in the multi-label decision list (Section 4.2.2).

Despite its appealing simplicity at a first glance, many different aspects of this approach have to be considered. As in traditional binary or multiclass problems, we need an approach for finding and evaluating rules in the multi-label setting. Basically, we use a modified version of Ripper [9] (cf. Section 4.4) which returns us the best rule for each label and selects the best among them with a separate selection heuristic (cf. Section 4.5) as proposed by Stecher et al. [43]. Selecting the best rule is, however, not trivial, since the labels may have already been covered to different degrees.

In the following, we start by introducing multi-label decision lists (Section 4.1, a type of rule sets that are capable of classifying multiple labels. We then give a more detailed description of the proposed approach including the learning and classification phase and provide subsequently a discussion on the arising issues as well as our ideas of how to deal with them in Section 4.2. Then, we present a toy example that highlights how a multi-label decision list learned by the proposed

iterative separate-and-conquer algorithm can be used to classify a multi-label instance (Section 4.3. In the remainder of the section, we illustrate how a single rule is learned (Section 4.4) and which kind of heuristic we used for doing so (Section 4.5).

4.1 Multi-label Decision Lists

With the SBR approach in Section 3.1 we induce two sets of decision lists $\mathcal{R}_i$ and $\mathcal{R}_i^*$, the first ones being only dependent on instance features and the second ones possibly also use label features. However, in this section, we are interested in obtaining one single condensed model $\mathcal{R}$ in form of an ordered decision list. As already discussed in Section 3.3, having only one list, e.g., prevents circuits in the classification and may provide an additional notion of the relevance of the labels regarding the classification. However, the straight-forward approach of combining induced single-label rules, such as those included in $\mathcal{R}_i$ and $\mathcal{R}_i^*$, into one single classical decision list is not possible since the classification process would stop as soon as the first label is set. Therefore, we propose an extension referred to as *multi-label decision lists*.

Similarly as for classical decision lists, rules are organized in a sequence $\mathcal{R} = \langle r_1, r_2, \ldots \rangle$ reflecting the order in which they are checked. However, the difference is that the classification process does not stop when a test instance is covered by one of the rules but continues with the subsequent rule in the sequence. Exceptions are if for all of the labels a rule was already found (all labels were classified) or the rule covering the test instance is explicitly marked as a *stopping rule*, in which case the classification process would immediately terminate. This mechanism allows to assign several labels with a single multi-label decision list, but also to terminate the classification if required. Stopping rules are necessary especially if only positive head rules are induced in order to prevent a bias towards predicting too much labels as true. In addition, they are reflected in the separate-and-conquer learning process, as well as the whole classification process itself. These aspects as well as more details on the classification process (Figure 7) are treated in Section 4.2.2. An example of a multi-label decision list is given in Table 3.

A multi-label decision list could be considered as a generalization of a single-label decision list as induced by regular covering algorithms. The stopping rule annotation allows to enforce the same behavior when only one positive label is allowed. On the other hand, it obviously also adopts characteristics from unordered rulesets, where the predictions of all rules are combined.

4.2 Multi-label Iterative Separate-and-Conquer

We propose to adapt the classical separate-and-conquer, or, covering algorithm, which was successfully applied for rule induction in many cases, in order to learn rules for MLC classification problems. Our main motivation in the design of the proposed algorithm was to keep the changes to the separate-and-conquer algorithm as small as possible. However, the increase in the number of targets to be considered imposes changes especially concerning the concept of the covering status and the separate step, i.e., the removal of instances from the training set. For instance,

Require: New training example pairs $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$,
parameters θ, τ, heuristic h, targets $\mathcal{B}$ (either $\mathcal{B} = \{1\}$ or $\mathcal{B} = \{0, 1\}$),
whether using stopping rules, whether re-inserting fully covered examples
$\mathcal{T} = \{(\mathbf{x}_1, \hat{\mathbf{y}}_1), \ldots, (\mathbf{x}_m, \hat{\mathbf{y}}_m)\}$ with $\hat{\mathbf{y}}_i = (?, ?, \ldots, ?)$, $i = 1 \ldots m$
while $|\mathcal{T}|/m \geq \theta$ **do** ▷ until, e.g., 95% of examples covered
  $r \leftarrow findBestGlobalRule(\mathcal{B}, \mathcal{T})$ ▷ get best possible rule regardless the head
  add r to decision list $\mathcal{R}$
  $(\mathcal{T}, \mathcal{T}_{part}, \mathcal{T}_{full}) = getCoveredSets(r, \mathcal{T})$ ▷ separate $\mathcal{T}$ according covering by r
  $\mathcal{T}_{add} \leftarrow getReAddSet(\mathcal{T}_{part}, \mathcal{T}_{full})$ ▷ depending on user parameters
  if $\mathcal{T}_{add} = \emptyset$ **then**
    mark r as stopping rule ▷ only uncovered examples in $\mathcal{T}$ of next round
  else
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_{add}$ ▷ add also some covered examples, do not remove them
return decision list $\mathcal{R}$

Fig. 3 Training algorithm for the multi-label iterative separate-and-conquer algorithm

Require: Example pairs $\mathcal{T}$, targets $\mathcal{B}$
$r \leftarrow \emptyset$, $r.h \leftarrow -\infty$ ▷ init best rule and its heuristic value
for each label y_i and target $t_{0/1} \in \mathcal{B}$ **do**
  $\mathcal{T}^i \leftarrow \mathcal{T}$
  remove all $\mathbf{x}$ where $\hat{y}_i \neq ?$ from $\mathcal{T}^i$ ▷ do not consider $\mathbf{x}$ if label already set
  $r' \leftarrow findBestRule(y_i, t_{0/1}, \mathcal{T}^i)$ ▷ find best body for target $y_i = t_{0/1}$
  $r'.h = h(r', y_i, \mathcal{T}^i)$ ▷ heuristic value depends on target label and $\mathcal{T}^i$
  if $r'.h > r.h$ **then**
    $r \leftarrow r'$ ▷ replace by better rule
return best rule r

Fig. 4 Algorithm *findBestGlobalRule* for finding the best current rule on the training set for any possible label in the head.

Require: Rule r, example pairs $\mathcal{T}$
$\mathcal{T}_{part} \leftarrow \emptyset, \mathcal{T}_{full} \leftarrow \emptyset$
for each example $(\mathbf{x}, \hat{\mathbf{y}}) \in \mathcal{T}$ **do** ▷ compute covering status for each example
  if r covers $\mathbf{x}$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \backslash \mathbf{x}$ ▷ remove since it may not be re-added
    apply head of r on $\hat{\mathbf{y}}$ ▷ replace corresponding value in $\hat{\mathbf{y}}$ if it was unset
    if $\hat{\mathbf{y}}$ is fully set **then** ▷ depending on $\mathcal{B}$, consider also unset zeros
      $\mathcal{T}_{full} \leftarrow \mathcal{T}_{full} \cup \mathbf{x}$
    else
      $\mathcal{T}_{part} \leftarrow \mathcal{T}_{part} \cup \mathbf{x}$
return uncovered ($\mathcal{T}$), partially ($\mathcal{T}_{part}$) and fully covered ($\mathcal{T}_{full}$) training examples

Fig. 5 Algorithm *getCoveredSets* for computing the covering status of examples for a given rule.

the question arises whether covered instances are labeled and re-added to the training set so that they may serve as an anchor point for following rules. The main difference to the original version of the algorithm clearly lies in its *iterative* nature. In addition, depending on whether we want to learn only positive heads or also negative ones, the definition of whether a label or an instance is covered or not changes.

The following subsections describes the training phase with its multiple aspects, followed by the classification phase.

Require: Partially and fully covered examples $\mathcal{T}_{part}, \mathcal{T}_{full}$,
parameter τ, whether using stopping rules, whether re-inserting fully covered examples
if use stopping rules **then**
  if full coverage rate $|\mathcal{T}_{full}|/(|\mathcal{T}_{full}| + |\mathcal{T}_{part}|) \geq \tau$ **then** ▷ e.g. 90%
    $\mathcal{T}_{add} \leftarrow \emptyset$ ▷ do not re-add any example although $\mathcal{T}_{part}, \mathcal{T}_{full}$ non empty
  else ▷ too many partially covered examples
    $\mathcal{T}_{add} \leftarrow \mathcal{T}_{part}$ ▷ re-add partially covered examples
    if re-insert all covered examples **then**
      $\mathcal{T}_{add} \leftarrow \mathcal{T}_{add} \cup \mathcal{T}_{full}$ ▷ re-add also fully covered examples
else
  $\mathcal{T}_{add} \leftarrow \mathcal{T}_{part}$ ▷ no stopping rules: re-add partially covered examples
return partially or fully covered examples $\mathcal{T}_{add}$ to be added again to training set

Fig. 6 Algorithm *getReAddSet* deciding wether covered examples are re-added to the training set.

Require: Test example $\mathbf{x}$, multi-label decision list $\mathcal{R}$
$\hat{\mathbf{y}} = (?, ?, \ldots, ?)$
for each rule r in rule list $\mathcal{R}$ **do** ▷ in the order of insertion
  if r covers $\mathbf{x}$ **then**
    apply head of r on $\hat{\mathbf{y}}$ if corresponding value in $\hat{\mathbf{y}}$ is unset
    if r marked as stopping rule **or** $\hat{\mathbf{y}}$ is complete **then**
      assume all remaining labels in $\hat{\mathbf{y}}$ are negative
      return $\hat{\mathbf{y}}$
assume all remaining labels in $\hat{\mathbf{y}}$ are negative
return $\hat{\mathbf{y}}$

Fig. 7 Application of a multi-label decision list to a test example.

4.2.1 Training

Figure 3 shows the pseudo-code for training a rule learner that is able to use predicted labels already in the learning process. The algorithms keeps track of two representations of the label vector:

- one for the original labels $(\mathbf{y}_1, \ldots, \mathbf{y}_m)$ and
- one for the labels that are currently accessible by the learner $(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m)$.

We initialize the label features $\hat{\mathbf{y}}_i$ with $(?, ?, \ldots, ?)$ since in the beginning no label is predicted yet.

The outer loop of the learning algorithm runs until only θ examples are left uncovered (Figure 3, line 2). It starts by searching for the best rule for each label (Figure 4). We consider two types of rules, either positive ($\mathcal{B} = \{1\}$) or positive and negative ($\mathcal{B} = \{0, 1\}$) head rules (cf. Section 2.3). Hence, for each label y_i the *findBestRule* (cf. Section 4.4) method is called with the current label y_i, an indication which type of head should be learned (either 0 or 1), and the current set of examples $\mathcal{T}^i$. Note that previously all examples $\mathbf{x}$ are removed for which the label at hand is already set by previous rules. The label is covered by now and does not have to be covered again. Note also that if these examples are not removed, *findBestRule* would often find the same rule again and again since the set of instances in $\mathcal{T}$ will often not change from one iteration to the next.

Instead of having access to label features that are initialized by, e.g., a BR approach as in the previous algorithm (cf. Section 3), we rely only on labels set by previous rules. Hence, it may take some time until enough rules are learned and,

consequently, enough labels are set so that they can be exploited in subsequent rule bodies.

After the best label-specific rule is found, its heuristic value is computed by using the coverage statistics derived on the original label at hand. Among these rule candidates (one for each label and target combination), we select the best one and add it to the decision list $\mathcal{R}$. Then, the rule is used to predict and set the i-th label feature $\hat{y}_i$ for all covered examples (Figure 5). To do so, we first remove the covered examples from $\mathcal{T}$ and for each example check whether all labels $\hat{\mathbf{y}}$ are already set. Depending on whether positive and negative heads or only negative heads are desired, we differentiate between checking if $\hat{y}_i \neq ?$ for $i = 1, \ldots, n$ or only for positive labels $y_i = 1$, respectively. If so, we store the example in $\mathcal{T}_{full}$. If labels are still missing we store the example in $\mathcal{T}_{part}$.

Re-inclusion of covered examples and usage of stopping rules We propose three different ways of re-adding covered examples to $\mathcal{T}$ (Figure 6), namely to re-include either

- only partially covered examples,
- partially and fully covered examples, or
- none (examples were almost fully covered).

In the first case, we remove all fully covered examples entirely from the training process and only add the partially covered instances to $\mathcal{T}$. However, this might introduce inconsistencies, since if we re-classify $\mathcal{T}_{full}$ with the final model, the current rule would apply as desired, but also possibly *any* following rule in the final decision list. These remaining rules are unlikely to be consistent with $\mathcal{T}_{full}$ since $\mathcal{T}_{full}$ was not used anymore for learning them. Moreover, it may be desirable to leave the fully covered examples in the training process since they may serve as an anchor point for subsequent rules and facilitate in such a manner the rule induction process.

Hence, we consider the option of marking rules as stopping rules (Section 4.1), which means that whenever the rule fires during classification this should stop the classification process since it is assumed that the prediction is already complete. This is considered to be the case if the percentage of fully covered examples among all covered examples is greater than a parameter τ (usually close to 1). If this happens, *getReAddSet* returns an empty set and r is marked as stopping rule (Figure 3, line 8). However, if the amount is below τ, the partially covered examples are added to the training set. Furthermore, we can set another parameter determining whether or not we would also like to re-include the fully covered examples in order to benefit from the additional information as mentioned above.

4.2.2 Classification

Figure 7 shows the pseudo-code of the classification process. Essentially, a test instance passes through the same steps as in the training process. We iterate over each single rules in the multi-label decision list and determine if it fires, i.e., if the rule covers the given test instance. Similarly as during training, we apply the predicted head label of the rule so that it can be checked by subsequent rules in the loop. However, if the rule was marked as stopping rule, the prediction is assumed to be complete and the process comes to an end.

Table 2 Extended multi-label WEATHER dataset.

outlook	temperature	humidity	windy	play	icecream	tea	lemonade	dontplay
rainy	65	70	yes	0	0	1	0	1
rainy	71	91	yes	0	0	1	0	1
sunny	85	85	no	0	1	0	1	1
sunny	80	90	yes	0	1	0	1	1
sunny	72	95	no	0	1	0	1	1
sunny	69	70	no	1	0	0	1	0
sunny	75	70	yes	1	0	0	1	0
overcast	83	86	no	1	0	0	1	0
overcast	64	65	yes	1	0	0	1	0
overcast	72	90	yes	1	0	0	1	0
overcast	81	75	no	1	0	0	1	0
rainy	70	96	no	1	0	0	1	0
rainy	68	80	no	1	0	0	1	0
rainy	75	80	no	1	0	0	1	0

Table 3 Learned rules on the WEATHER dataset with the proposed multi-label iterative separate-and-conquer approach.

$icecream \leftarrow$ outlook = sunny, humidity ≥ 85
$\overline{icecream} \leftarrow \emptyset$
$tea \leftarrow$ outlook=rainy, windy=yes
$tea \leftarrow \emptyset$
$lemonade \leftarrow \overline{tea}$
$play \leftarrow \overline{icecream}, \overline{tea}$
$\overline{dontplay}^* \leftarrow play$
$\overline{play} \leftarrow \emptyset$
$dontplay \leftarrow \emptyset$
$\overline{lemonade}^* \leftarrow \emptyset$

We make two important assumptions about the classification. Firstly, we assume that the first prediction for a particular labels is the valid one and cannot be revoked afterwards. This is also ultimately reflected in the training process. However, it makes the approach more sensitive to error propagation than the common decision lists since rules may depend on the correct assignments of the previous rules in the decision list. A possible alternative would be to consider several assignments for the same label, e.g., increasing or decreasing the evidence or confidence for a label decision. On the other hand, aggregating and weighting several decisions is not trivial. We leave the analysis of the possible solutions for further work.

Secondly, it is common in MLC to assume that a label is irrelevant if not determined otherwise. This assumption follows from the usual asymmetry of set and unset labels in common multi-label problems. Hence, we predict a label as irrelevant if it was not covered by any rule. Future versions of our approach could consider to differentiate w.r.t. every label separately. However, in case of learning positive and also negative head rules, the learning algorithm itself takes care of considering the asymmetry by adding corresponding rules (see empty rules in Table 3). Comparing to traditional decision lists, our assumption corresponds to *the default rule* which usually predicts the majority class.

4.3 Toy Example

The following toy example, which is based on the well known WEATHER dataset, illustrates the functioning of the proposed approach (Table 2). The dataset indicates the decision whether to play tennis depending on the weather forecast. Basically, we should *play* tennis if it is overcast, or if it is sunny and normally humid, or if it is rainy but not windy. We introduce labels for indicating that we should eat *ice cream* if it is sunny and too humid and drink *tea* if it is rainy and windy. We should drink *lemonade* if it is warm enough. The label *dontplay* is included for illustrative purposes. Obviously, several dependencies are introduced, such as negation, mutual exclusion and subsumption.

In Table 3, the rules are shown as learned by the multi-label iterative separate-and-conquer approach. They are represented in a multi-label decision list. Rules that are marked with a "*" are stopping rules that terminate the classification. The following test example might illustrate how the classification process works: $(rainy, 80, 75, no,\ ?, ?, ?, ?, ?, ?)$.

As the first rule in the list does not cover the example, the second rule in the list is checked. This rule always evaluates to true and hence *icecream* is set to 0. As described in Section 4.2.2, the label of the instance is set accordingly, resulting in $(rainy, 80, 75, no,\ ?, 0, ?, ?, ?)$. Again, the third rule does not cover the example and rule number four sets $tea = 0$. Then, rule number five changes the instance to $(rainy, 80, 75, no,\ ?, 0, 0, 1, ?)$. The sixth rule sets $play = 1$ since $icecream = 0$ and $tea = 0$. In the end, *dontplay* is then set to 0 by rule number seven and the classification is terminated as this is a stopping rule.

4.4 Finding the Best Label-Specific Rule with Ripper

In preliminary evaluations, we used a simple SeCo algorithm which is based on a top-down hill-climbing search. Albeit the heuristic of this algorithm is tuned to be suited for a variety of different requirements [23], it turned out that without proper rule pruning phase, this leads to a large set of rules. Moreover, our results on the stacking approach suggest that explicit pruning is particularly useful in order to emphasize on and make use of dependencies between labels (see Section 6.3). Additional reasons for using the Ripper algorithm are the efficient implementation available in Weka [51] and the better comparability to the stacking approaches especially w.r.t. the found models. Hence, we provide in the next subsection a brief description of the algorithms and our modifications.

Ripper is based on the IREP-strategy [17], hence, in the beginning the dataset is divided into a *growing* and a *pruning* set where the first usually contains $\frac{2}{3}$ of the training set and the rest of the instances are included in the latter set. A rule is learned on the growing set by employing the FOIL-gain heuristic that relates the quality of the current refinement of a rule to that of its predecessor. By successively selecting the best possible condition in a greedy hill-climbing way, the algorithm proceeds with refining a rule as long as it covers negative examples. Hence, a rule learned by this procedure is consistent, i.e., it does not cover any negative examples. That rule then is immediately pruned on the pruning set by successively removing the last condition as long as the heuristic value $\frac{p-n}{p+n}$ does not decrease. In the basic version of how we used Ripper, *findBestRule* stops at

this point and returns the pruned rule as we are only interested in the single best rule per label.

However, this skips the special optimization phase of Ripper which revises the found rules in the context of the remaining rules in the decision list after the final ruleset is learned. Since subsequent rules are unknown during the learning of a single rule, they may be suboptimal in the final decision list. Hence, after finding the first best rule, the covered examples are removed from the training set and the process is repeated until no positive examples are left or a minimum description length (MDL) stopping criterion is met. Ripper then continues with its post-optimization. Each rule in the resulting ruleset is inspected, and if the MDL criterion is met, it is substituted by a *replacement* rule learned from scratch or by a *revision* of the original rule.

Certainly, revising the ruleset in the context of all rules is one of the key reasons for the good performance of Ripper apart from using IREP. In order to also benefit from this optimization and in view of lacking a proper adaptation to multi-label decision lists, we consider to simply adopt the first rule left after learning the whole ruleset for the given label and after applying the optimization phase. The assumption, that the same remaining ruleset may be learned again in the next rounds (i.e., in the next execution of *findBestRule* for the current label), is admittedly not often valid. In fact, it is only true in any case if we assume independent labels, since then the multi-label SeCo algorithm would correspond to learning separate rulesets for each label and then joining them. On the other hand, it may also be valid to a certain degree when using the option of re-adding full covered examples. And certainly, it is not less reasonable than learning rules independently of any following rule when not using the post-optimization. However, we prefer to consider the usage of the Ripper optimization as an additional pruning heuristic and not as a substitution for a proper post-optimization of multi-label decision lists. This step is left for future work.

4.5 Heuristic for Selecting the Best Rule

For selecting the final rule among all the best rules per label, we experimented with different heuristics. The main trade-off comes from two objectives:

Coverage optimization: The more examples are covered, the more label features are set and the better the subsequent rules may utilize these label features.

Consistency optimization: The covered examples should also be covered correctly as subsequent rules may rely on the classified feature labels.

Hence, we used heuristics that are able to trade off these two objectives. As our main evaluation metrics are precision, recall, and their combination, the F-measure, a natural choice was to also employ this heuristic for rule selection. The F-measure is defined by

$$h_F = \frac{\left(\beta^2 + 1\right) \cdot h_{prec} \cdot h_{rec}}{\beta^2 \cdot h_{prec} + h_{rec}} \tag{1}$$

where $h_{prec} = \frac{p}{p+n}$ and $h_{rec} = \frac{p}{P}$.

Note that in our setting, the compared rules usually have different P and N due to the adapted training sets $|\mathcal{T}_i| = P + N$. The F-measure takes this into

consideration, e.g., by assigning a higher weight to rules which have the same precision but apply to a higher number of instances. It is also ensured by this computation that infrequent labels are not under-represented.

We experimented with the best parameter settings used by Janssen and Fürnkranz [23] and also with different settings for β. Preliminary evaluations showed that $\beta = 1$ was a reasonable choice for our purposes, also in consideration of our main multi-label evaluation measure, so that we used this setting for all our subsequent experiments. However, it seems worthwhile to take a closer look on the choice of evaluation heuristics and our work can only be considered as a first step towards understanding the usage of heuristics in the special case of multi-label classification.

5 Related Work

In the following, we revise the most relevant literature on modeling and exploiting label dependencies, making dependencies explicit and learning rules for multi-label classification. Due to the vast amount of work on MLC, we refer the reader to the excellent surveys available for a more complete overview [46, 48, 53, 19]. An extensive comparative study of state-of-the-art approaches is provided by [33].

5.1 Label Dependencies and Stacking

Stacking label features and label predictions is an important technique in order to exploit label dependencies, especially conditional dependencies, and was already discussed together with the most relevant literature in Section 3 and 3.1. Further works include probabilistic CCs [11], which uses probabilistic base classifiers in CC for exhaustively explore the joint distribution. Read et al. [39] and Kumar et al. [25] use Monte Carlo and beam search, respectively, in order to approximate the joint mode. The latter work uses kernel target alignment to order the chain according to the difficulty of the single-label problems. Sucar et al. [44] perform an analysis of the global dependencies by using a Bayesian net in order to determine a reasonable order of the labels. All these method have in common with the original CC that they can only effectively exploit one direction of a dependency between two labels. Li and Zhou [27] propose to filter the CCs of an ensemble of CCs [40] in order to maximize F1. Unfortunately, no post-analysis is performed of the useful pairing directions.

Stacking approaches not related to CC include the method by Cheng and Hüllermeier [8], which uses k-NN in order to stack the number of occurrences of labels in the neighborhood as new features. Li and Zhang [28] exclude certain labels from the stacking if their respective base classifiers underperform on a validation set. Similarly to Tsoumakas et al. [47], they then select a fixed number m of useful labels for each respective label based on a features subset selection approach. Subsequently, an ensemble of m classifiers are trained for each label adding one respective stacked label feature to each of the binary datasets. In our proposed stacking approach, noisy labels are implicitly excluded by the underlying rule learning approach. However, explicit filtering techniques could support this

selection process especially for small amounts of data. Madjarov et al. [31, 32] introduce different methods of stacking predictions for the pairwise learners of a more efficient two-stage variant of the calibrated label ranking approach of Fürnkranz et al. [16]. Huang et al. [22] use a boosting approach which is modified so that the models for each of the labels are allowed to use the decision stumps of the other models. By measuring how often this happens, it is possible to also determine a degree of relationship. Alessandro et al. [3] construct a Bayesian net by using a similar approach to stacking, namely by learning a set of single-label Bayes nets depending on all other labels and a set depending on all other labels and the features. These nets are then concatenated so that one Bayes model for each of the labels is obtained.

Stacking label features is a rather new direction for exploiting dependencies. Early works on exploiting dependencies date back to 1999, when McCallum [34] produced generative models for labelsets from mixtures of topic based word distributions. Ghamrawi and McCallum [18] used conditional random fields parameterized by label co-occurrences and Zhu et al. [54] proposed a label correlations conditioned maximum entropy method. Usually, these approaches only considered unconditional dependencies, or at least did not differentiate between them. An exception is, e.g., Zhang and Zhang [52], who proposed to use a Bayesian network where the nodes are conditioned on the instance features. In addition, they proposed to categorize dependencies into first, second, and high-order degree according to the number of labels involved.

Recently, Chekina et al. [7] analyzed the different types of dependencies between pairs of labels on the standard benchmark datasets. Unconditional dependencies were analyzed by a simple χ^2 test on the label co-occurrence matrix whereas for detecting unconditional dependencies they compared the performance of a classifier h_i for a label y_i trained on the instance features ($\mathbf{x}$) to the same learning algorithm being applied to the input space $(\mathbf{x}, y_j)$ augmented by the label feature of a second label y_j. If the predictions differ statistically significantly, then y_i is assumed to be conditionally dependent on y_j. Their evaluations show that much more labels in common benchmark datasets are pairwise unconditionally dependent than pairwise conditionally dependent, and that, surprisingly, modeling global dependencies is more beneficial in terms of predictive performance. However, this statement is very specific to their setting. The dependency information is basically used in order to guide a decomposition into smaller problems with less labels which are either independent or dependent. In contrast to our methods, only pairwise co-occurrence and pairwise exclusion can effectively be exploited by their approach. Pairwise subset (implication) and exclusion constraints were already discovered by the approach of Park and Fürnkranz [37] by using an association rule miner (Apriori). These were later used in the classification process in order to correct predicted label rankings which violated the globally found constraints. Surprisingly, the correction did not reveal any significant impact on the ranking performance.

Very recently, Papagiannopoulou et al. [36] proposed a method for discovering deterministic positive entailment (implication) and exclusion relationships between labels and sets of labels. The base relationships are extracted with an association rule miner and represented in a deterministic Bayesian network with the help of virtual leak nodes. The marginal probabilities produced by a base multi-label classifier are then corrected by probabilistic inference, improving ranking performance

on a variety of datasets. More interestingly, the authors use their method to extract and analyze relationships in the used benchmark datasets. Those relationships are implicitly used in the inference process, in contrast to our approaches, which produce actual classification rules. Moreover, their approach focuses on global dependencies only. A further limitation is the sensitivity to the minimum support parameter in the used association rule mining algorithm (Apriori) which leads to a high variability in the number of discovered relationships. Nevertheless, especially the explicit notation of sets of exclusive labels and the resolution of circular dependencies (cf. Section 3.3) are advantages which reveal interesting lines of future work for our proposed approaches.

5.2 Rule Learning for Multi-label Classification

Many rule-based approaches to multi-label learning rely on association rules. As the goal is classification, usually classification association rules (CARs) are used, which only have label features in the rule heads, instead of regular association rules that would also find relations between instance features. Approaches using CARs were already successfully used for multiclass classification. Liu et al. [29], for instance, use classical association rule mining in order to obtain a candidate set of CARs for their classification system based on associations (CBA). The candidates, which are ordered according to their confidence, are then iteratively processed and included in the final ruleset if they cover at least one of the remaining uncovered training instances. Their system outperformed C4.5 decision trees and C4.5 generated decision rules.

Ávila et al. [5] adapted the idea to multi-label learning. The authors use a genetic algorithm to induce single-label association rules. A multi-label prediction is then built by using a combination of all covering rules of the BR rulesets. A good distribution of the labels is also ensured by using a token-based re-calculation of the fitness value of each rule. The approach of Li et al. [26] learn single-label association rules as well. For prediction, exactly those labels are set that have a probability greater than 0.5 in the covering rules.

A different idea is to introduce multi-label instead of single-label head rules. Those are able to directly classify a multi-label instance without the need to combine single-label rules [4]. However, the classification is then done by using a weighted voting scheme as many multi-label rules may cover the example. Another associate multi-label rule learner with several possible labels in the head of the rules was developed by Thabtah et al. [45]. The approach transforms all frequent itemsets passing a certain minimum confidence threshold into single-label head rules. Rules with the same body are merged into multi-label head rules. The ruleset is extended in subsequent iterations by repeating the procedure on uncovered examples. Ženko and Džeroski [50] present an approach for inducing predictive clustering rules (PCR) for multi-target problems. Note that multi-label classification can be considered as a special case of multi-target classification with only binary targets. Their approach is based on the separate-and-conquer principle, but uses evaluation heuristics better known from clustering. More specifically, a rule is evaluated according to the dispersion among the covered examples (the cluster), which is the averaged deviance of their target attribute values from the cluster prototype, i.e., from the multinomial attribute distribution vector. The

heads of the rules are the associated cluster prototypes. Hence, applied on multi-label data this approach would obtain (label-independent) multi-label head rules. The approach was also extended to multi-target regression by generating ensembles of multi-target head rules from learned predictive clustering regression trees [2, 1].

The reviewed approaches induce classification rules from the whole training set during the training phase. The multi-label lazy associative approach of Veloso et al. [49] in contrast generates CARs from the neighborhood of a test instance during prediction. The advantage is that fewer training instances are used to compute the coverage statistics which is beneficial when small disjuncts are a problem as they are often predicted wrongly due to whole training set statistics. Similarly to our multi-label iterative separate-and-conquer approach, they propose a variant which uses the prediction of a first classification iteration in order to enhance the second iteration. The corresponding predicted label is used in order to filter a different set of nearest neighbors.

The multi-label iterative separate-and-conquer approach adopt certain concepts from self-training [6, 41, 14]. Here, we have a pool of labeled and unlabeled data. A classifier is then learned on the labeled data and used to classify a number of instances of the unlabeled pool. After that, these instances are added to the labeled data pool and the procedure is repeated. The unset feature labels in our iterative approach can be seen as the unlabeled pool. As each learned rule is used immediately after it was learned in order to label the covered instances and these are re-included into the training set, the next training iteration relies on the classification of the previous one just as in self-training.

6 Evaluation

In the evaluation, the main focus lies on the inspection and the analysis of the induced rule models. Statistics about the revealed dependencies are discussed and exemplary rule models are shown. The two proposed approaches are also compared against each other and to a multi-target rule learner in terms of predictive performance assessed by several multi-label measures.

6.1 Setup

An overview of the used datasets[2] is given in Table 4. They are from different domains and have varying properties. Details of the data are given in the analysis when needed.

As a rule learner, we used the JRip implementation of Ripper [9] from Weka [51] with default parameters except for the following settings. Depending on the experiment, we turn pruning on or off and execute zero or two optimization phases. Ripper usually produces rules only for the minority class(es). In order to obtain positive and negative heads for the decomposition approaches, we basically applied the iterative multi-label SeCo approach on the resulting binary subproblems. We

[2] We refer to the MULAN repository for details and sources: http://mulan.sf.net/datasets.html.

Table 4 Statistics of the used datasets: name of the dataset, domain of the input instances, number of instances, number of nominal/binary and numeric features, total number of unique labels, average number of labels per instance (cardinality), average percentage of relevant labels (label density), number of distinct labelsets in the data.

name	domain	instances	nominal	numeric	labels	cardinality	density	distinct
EMOTIONS	music	593	0	72	6	1.869	0.311	27
SCENE	image	2407	0	294	6	1.074	0.179	15
YEAST	biology	2417	0	103	14	4.237	0.303	198
GENBASE	biology	662	1186	0	27	1.252	0.046	32
MEDICAL	text	978	1449	0	45	1.245	0.028	94
ENRON	text	1702	1001	0	53	3.378	0.064	753
MEDIAMILL	video	43907	0	120	101	4.376	0.043	6555
COREL16K (set 1)	images	13766	500	0	153	2.859	0.019	4803
BIBTEX	text	7395	1836	0	159	2.402	0.015	2856
CAL500	music	502	0	68	174	26.0	0.150	502
COREL5K	images	5000	499	0	374	3.522	0.009	3175

differentiate both variants with the symbols + and ±. As a basis for our framework, we used the SeCo-framework for rule learning [24].

Further details on the employed algorithms, combinations of algorithms and notations used are given in the following:

- We refer to the binary relevance decomposition as BR. It is either used with the normal Ripper variant (BR^{+}) or with the two-targets one ($BR^{\pm}$). Except for the model analysis experiments, pruning and optimization was turned on.
- The stacked variant of binary relevance (SBR) was either parameterized in order to induce positive head rules (SBR^{+}) or positive and negative ones ($SBR^{\pm}$). The same parameters were used for the underlying BR classifiers. For certain analyses, we also employed a variant which only used label features $\mathbf{y}$ as inputs, referred to as SBR_y, instead of training on $\mathbf{x}$ and $\mathbf{y}$ (SBR_{xy} or without subscripts). If not indicated otherwise, the latter variant is used. $SBR_?$ and SBR_d refer to whether predicting the default label is interpreted as abstention or as negative prediction, respectively. The number of bootstrapping iterations was set to $m = 10$.
- SBR_{Gibbs} denominates the approach without additional BR classifiers but with sampling random initializations. We set the number of samples to 1000 and the number of burn-in iterations to 100, as proposed by Guo and Gu [21]. However, in our experiments the predictions always converged in less than 10 burn-in iterations.
- Multi-label iterative separate-and-conquer is indicated with SeCo. Positive head rules are learned by $SeCo^{+}$, whereas $SeCo^{\pm}$ learns both types. $Seco_{opt}$ indicated the usage of two optimization phases. Based on preliminary evaluations on EMOTIONS, we set $\theta = 0.01$, $\tau = 0.1$ and the option of readding fully covered examples for all datasets without further adaptations. SeCo did not finish on time on the datasets with more than 5000 instance features due to the sensitiveness of the used Ripper implementation regarding the dimensionality of input features. Note that Ripper is executed each time a rule candidate is generated. Therefore, we do not report results for these four datasets for SeCo.
- For comparison, we included results from the predictive clustering rules algorithm (PCR), which learns dense multi-label head rules (cf. Section 5.2). We used the standard covering approach producing ordered rulesets and the multiplicative version of the dispersion heuristic, which consistently gave the best

results, is computed on the instance features only.[3] Parameters not mentioned were set to the default values.

6.2 Evaluation Measures

We use a wide range of different multi-label bipartition measures in order to evaluate our results, which we will introduce in the following. Note that the used algorithm do not produce any label rankings.

For the evaluation with micro-averaged recall, precision and F1, we computed a two-class confusion matrix for each label ($y_i = 1$ vs. $y_i = 0$) and eventually aggregated the results by (component-wise) summing up all n matrices into one global confusion matrix (cf. [48]). Recall and precision is computed based on this global matrix in the usual way, F1 denotes the unweighted harmonic mean between precision and recall (cf. Eq. 1). More formally, for predictions $\hat{\mathbf{y}}_i = (\hat{y}_{i,1}, \ldots, \hat{y}_{i,n})$ and true label vectors $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,n})$, $i = 1 \ldots m$, micro recall and precision are given by

$$\text{Mi. Recall} : \frac{\sum_i |\mathbf{y}_i \cap \hat{\mathbf{y}}_i|}{\sum_i |\mathbf{y}_i|} \quad \text{Mi. Precision} : \frac{\sum_i |\mathbf{y}_i \cap \hat{\mathbf{y}}_i|}{\sum_i |\hat{\mathbf{y}}_i|}$$

where $|\mathbf{y}_i \cap \hat{\mathbf{y}}_i| = \sum_j [[y_{i,j} = \hat{y}_{i,j} = 1]]$, $|\mathbf{y}_i| = \sum_j [[y_{i,j} = 1]]$, $j = 1 \ldots n$ going over the labels and $[[z]]$ returns 1 if z is true, otherwise 0.

For the macro-averaged measures, recall and precision are computed on the label-wise confusion matrices and these values are then averaged. More formally, we compute the metrics by

$$\text{Ma. Recall} : \frac{1}{n} \sum_j \frac{\sum_i [[y_{i,j} = \hat{y}_{i,j} = 1]]}{\sum_i [[y_{i,j} = 1]]} \quad \text{Ma. Precision} : \frac{1}{n} \sum_j \frac{\sum_i [[y_{i,j} = \hat{y}_{i,j} = 1]]}{\sum_i [[\hat{y}_{i,j} = 1]]}$$

Macro F1 uses the recall and precision values for each label (the fractions in the equation before) to compute the harmonic means. These values are then averaged over the labels.

In addition, we report the Hamming accuracy, which is essentially the average accuracy of each label prediction, and subset accuracy, which is the rate of perfectly classified instances:

$$\text{Hamming Acc.} : \frac{1}{m \cdot d} \sum_i \sum_j [[y_{i,j} = \hat{y}_{i,j}]] \quad \text{Subset Acc.} : \frac{1}{m} \sum_i [[\mathbf{y}_i = \hat{\mathbf{y}}_i]]$$

The measures were averaged over the ten-fold cross validation results, which we used for all the performance experiments.

For common multi-label problems, it is usually more interesting and important to correctly classify relevant labels than irrelevant ones. This aspect is especially reflected by the recall, precision and F1 measures, which completely ignore true negatives. Table 2 may serve for further illustration. A classifier concerned with maximizing F1 would especially try to match the 1's in the label matrix on the right. Roughly speaking, micro- and macro-averaging differ in that matching a 1 (a relevant label) in a column (label) with only few 1's becomes more important than

[3] We use the implementation of CLUS, available at https://dtai.cs.kuleuven.be/clus/.

Table 5 Statistics of the non-stacked and stacked BR models with pruned or unpruned and positive and positive or negative heads versions. From left to right, for the BR model: (1) avg. # rules per label, (2) avg. # conditions per rule, (3) avg. # conditions per label. For the stacked model: (4) avg. # rules per label, (5) avg. # conditions per rule, (6) avg. # conditions per label, (7) percentage of conditions with label feature tests, perc. of rules depending (8) *only* on label features, (9) partially, (10) *only* on instance features, perc. of label rulesets depending (11) *only* on label features, (12) partially, (13) *only* on instance features.

dataset /	BR: rules & cond.			SBR: rules & cond.			cond.	rule stats			label rulesets stats		
approach	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
EMOTIONS													
pruned/+	4.00	2.92	11.67	3.00	2.89	8.67	36.54%	5.56%	61.11%	33.33%	16.67%	83.33%	0.00%
unpruned/+	12.00	4.11	49.33	12.33	4.30	53.00	16.98%	0.00%	48.65%	51.35%	0.00%	100.00%	0.00%
pruned/±	3.83	1.61	6.17	2.67	1.69	4.50	51.85%	37.50%	25.00%	37.50%	0.00%	100.00%	0.00%
unpruned/±	14.83	3.51	52.00	14.17	3.21	45.50	10.99%	0.00%	31.76%	68.24%	0.00%	100.00%	0.00%
SCENE													
pruned/+	7.33	4.39	32.17	5.00	4.83	24.17	17.93%	0.00%	46.67%	53.33%	0.00%	100.00%	0.00%
unpruned/+	14.83	5.42	80.33	12.00	5.10	61.17	10.35%	0.00%	29.17%	70.83%	0.00%	100.00%	0.00%
pruned/±	8.00	2.27	18.17	7.67	2.59	19.83	22.69%	13.04%	28.26%	58.70%	0.00%	100.00%	0.00%
unpruned/±	21.33	4.04	86.17	13.83	3.72	51.50	12.94%	2.41%	26.51%	71.08%	0.00%	100.00%	0.00%
YEAST													
pruned/+	3.29	4.00	13.14	4.14	2.71	11.21	55.41%	43.10%	50.00%	6.90%	14.29%	85.71%	0.00%
unpruned/+	6.86	6.22	42.64	11.00	3.95	43.50	29.89%	14.29%	72.08%	13.64%	0.00%	100.00%	0.00%
pruned/±	5.21	1.95	10.14	3.71	1.54	5.71	75.00%	75.00%	3.85%	21.15%	57.14%	42.86%	0.00%
unpruned/±	43.86	4.96	217.57	11.29	3.43	38.71	22.69%	16.46%	33.54%	50.00%	7.14%	92.86%	0.00%
GENBASE													
pruned/+	0.93	1.08	1.00	0.93	1.04	0.96	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
unpruned/+	1.04	1.29	1.33	1.04	1.29	1.33	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
pruned/±	1.04	1.07	1.11	1.04	1.07	1.11	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
unpruned/±	1.19	1.16	1.37	1.15	1.16	1.33	2.78%	3.23%	0.00%	96.77%	0.00%	3.70%	96.30%
MEDICAL													
pruned/+	1.07	1.73	1.84	1.07	1.88	2.00	17.78%	0.00%	27.08%	72.92%	0.00%	20.00%	80.00%
unpruned/+	2.84	3.51	9.98	2.29	3.22	7.38	15.36%	0.00%	38.83%	61.17%	0.00%	28.89%	71.11%
pruned/±	1.16	1.40	1.62	1.24	1.34	1.67	17.33%	12.50%	7.14%	80.36%	0.00%	20.00%	80.00%
unpruned/±	2.82	2.67	7.53	2.38	2.19	5.20	11.11%	6.54%	14.95%	78.50%	0.00%	31.11%	68.89%
ENRON													
pruned/+	1.45	3.68	5.34	1.83	3.49	6.40	37.46%	15.46%	58.76%	25.77%	3.77%	54.72%	41.51%
unpruned/+	5.77	5.21	30.06	7.32	4.77	34.91	26.32%	1.80%	77.32%	20.88%	0.00%	88.68%	11.32%
pruned/±	1.66	1.81	3.00	2.06	1.81	3.72	47.21%	34.86%	11.93%	53.21%	18.87%	37.74%	43.40%
unpruned/±	19.49	4.20	81.87	13.64	3.80	51.87	19.46%	4.15%	48.55%	47.30%	1.89%	90.57%	7.55%
CAL500													
pruned/+	0.44	2.32	1.01	1.34	1.98	2.65	63.34%	49.36%	43.78%	6.87%	31.03%	46.55%	22.41%
unpruned/+	5.92	3.99	23.63	7.16	3.59	25.67	31.27%	2.89%	73.92%	23.19%	1.15%	97.70%	1.15%
pruned/±	0.99	1.35	1.34	1.87	1.23	2.30	68.75%	68.31%	7.69%	24.00%	48.28%	33.91%	17.82%
unpruned/±	11.34	3.14	35.64	8.68	2.88	25.02	26.60%	6.89%	46.29%	46.82%	2.87%	95.98%	1.15%
MEDIAMILL													
pruned/+	18.91	6.44	121.71	14.69	5.76	84.57	31.43%	5.53%	84.37%	10.11%	6.93%	93.07%	0.00%
unpruned/+	37.05	6.83	252.95	29.49	6.83	201.33	29.02%	2.12%	91.27%	6.62%	3.96%	96.04%	0.00%
pruned/±	7.12	3.17	22.58	4.02	2.31	9.29	46.38%	44.83%	29.31%	25.86%	46.53%	51.49%	1.98%
unpruned/±	47.32	5.78	273.51	44.83	6.76	303.08	14.52%	3.60%	49.09%	47.31%	20.79%	78.22%	0.99%
COREL16K													
pruned/+	2.44	2.14	5.22	4.33	2.82	12.22	64.37%	25.79%	72.10%	2.11%	11.76%	81.70%	6.54%
unpruned/+	6.80	3.27	22.24	10.43	4.54	47.35	52.20%	3.82%	94.42%	1.75%	0.65%	97.39%	1.96%
pruned/±	1.13	8.50	9.61	1.58	2.00	3.17	88.87%	88.84%	10.33%	0.83%	73.20%	12.42%	14.38%
unpruned/±	34.34	5.43	186.47	32.73	7.10	232.22	44.28%	10.35%	80.65%	9.01%	7.19%	92.81%	0.00%
BIBTEX													
pruned/+	4.87	3.07	14.98	5.12	2.89	14.77	18.31%	5.65%	36.73%	57.62%	1.89%	81.76%	16.35%
unpruned/+	16.40	3.98	65.30	16.64	3.83	63.66	11.61%	0.64%	36.03%	63.33%	0.00%	96.86%	3.14%
pruned/±	1.67	1.89	3.16	1.65	1.77	2.92	23.23%	14.45%	21.67%	63.88%	10.06%	34.59%	55.35%
unpruned/±	13.47	4.84	65.23	10.31	4.95	50.97	14.92%	1.59%	48.57%	49.85%	0.00%	91.19%	8.81%
COREL5K													
pruned/+	1.15	2.09	2.41	1.83	2.34	4.27	64.33%	33.67%	60.32%	6.00%	12.03%	50.80%	37.17%
unpruned/+	2.70	3.01	8.12	4.00	3.52	14.08	49.97%	8.09%	87.76%	4.15%	2.67%	72.19%	25.13%
pruned/±	2.70	3.01	8.12	0.99	1.77	1.76	83.92%	81.18%	14.78%	4.03%	53.21%	14.17%	32.62%
unpruned/±	9.39	7.26	68.17	7.33	5.01	36.74	46.94%	16.05%	71.26%	12.69%	20.05%	74.06%	5.88%

matching 1's in more dense columns for macro-averaging, whereas micro-averaging does not discriminate between the positive assignments.

We believe that our approaches do not make a special treatment to more or less frequent labels. Therefore, we consider that the micro measures are more suited in order to evaluate and compare our proposed methods to each other. Hence, our analysis mainly focuses on this family of measures.

6.3 Model and Data Analysis

The first block of columns (1-3) in Table 5 shows the statistics of the underlying BR decision lists, whereas the following blocks provide statistics for the stacked models on top. In contrast, Table 6 combines both models by grouping rules with the same label in the head. This table also includes the statistics for the models obtained by the multi-label iterative separate-and-conquer algorithm as well as the stacked model only depending on label features (SBR_y). For the model analyses, we obtained the rule models on the whole training sets, respectively.

6.3.1 Label Conditions

Column (7) shows the percentage of conditions on labels w.r.t. to all conditions in the model. We see that there is a great divergence between the datasets. E.g., the models for GENBASE do not use label features at all (except for the unpruned $\pm$ variant), i.e., their rules' bodies are completely label-independent. This is a strong indicator for only very weak dependencies in this dataset. This is remarkable, since this breaks the main assumption of MLC, and yet this dataset may have often been used in the literature to show the ability of a certain algorithm to exploit label dependencies. In the case of only weak dependencies though, learning each label independently is already sufficient and exploiting (possibly non-existing) label dependencies clearly will not yield much better performance. At least, our rule learners were not able to make any use of the additional 26 features in order to improve their performance. A view into columns (1)-(6), the prediction quality (Table 8) and eventually into the models, reveals that the presence of one single short amino acid chain (instance feature) is often enough to correctly predict a particular functional family (label). If we forced the rule learner to use only the label features (SBR_y^+ in Table 6), we could observe that a high number of label features is needed in average in order to reconstruct a target label. As it can be seen in the next section, this is reflected in a difference in the predictive performance, although the margin is interestingly not very pronounced. Apparently, the learner compensates missing low-level dependencies by a higher complexity of the model and hence high-level relations. The same observation can be made for MEDICAL. SBR_y^+ can only compensate the missing information by exploiting long combinations among the 45 labels.

For the multi-label iterative separate-and-conquer models, we observed that considerably less rules depend on labels than for the stacking approaches. The most obvious explanation for this behavior is the iterative and recursive methodology of the SeCo mechanism itself. Label features become available for subsequent rules only after they were predicted before by one of the rules. In fact, the amount of label-dependent rules and conditions is bounded by the number of rules of the model itself. As column (4) reveals, the number is usually small. In contrast, SBR disposes of the full label feature vector from the beginning. This is especially severe for small and difficult to learn labels and for the SeCo^+ variant. Due to the used rule learning heuristic, rules covering small labels or rules covering only a smaller subset of instances of labels that are hard to discriminate are more likely to be selected lately in the process. However, especially small labels are potentially very useful in order to predict other labels, since they tend to be more often enclosed by them.

Table 6 Combined statistics of the full models. Statistics for SBR include the statistics of the BR models used for initialization. (4)-(11) as in Table 5.

dataset	approach	rules & cond.			cond.	rule stats			label rulesets stats		
		(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
EMOTIONS	SBR^{+}_{xy}	7.00	2.90	20.33	15.57%	2.38%	26.19%	71.43%	0.00%	100.00%	0.00%
	$SBR^{\pm}_{xy}$	6.50	1.64	10.67	21.88%	15.38%	10.26%	74.36%	0.00%	100.00%	0.00%
	SBR^{+}_{y}	5.33	2.88	15.33	23.91%	25.00%	0.00%	75.00%	0.00%	100.00%	0.00%
	$SeCo^{+}$	3.67	2.36	8.67	1.92%	4.55%	0.00%	95.45%	0.00%	16.67%	83.33%
	$SeCo^{+}_{opt}$	3.83	1.70	6.50	7.69%	8.70%	4.35%	86.96%	0.00%	50.00%	50.00%
	$SeCo^{\pm}$	4.00	1.71	6.83	7.32%	8.33%	4.17%	87.50%	0.00%	50.00%	50.00%
	$SeCo^{\pm}_{opt}$	3.17	1.37	4.33	7.69%	0.00%	10.53%	89.47%	0.00%	33.33%	66.67%
SCENE	SBR^{+}_{xy}	12.33	4.57	56.33	7.69%	0.00%	18.92%	81.08%	0.00%	100.00%	0.00%
	$SBR^{\pm}_{xy}$	15.67	2.43	38.00	11.84%	6.38%	13.83%	79.79%	0.00%	100.00%	0.00%
	SBR^{+}_{y}	8.33	4.46	37.17	13.45%	12.00%	0.00%	88.00%	0.00%	100.00%	0.00%
	$SeCo^{+}$	7.83	3.04	23.83	1.40%	4.26%	0.00%	95.74%	0.00%	16.67%	83.33%
	$SeCo^{+}_{opt}$	7.00	2.74	19.17	1.74%	2.38%	2.38%	95.24%	0.00%	33.33%	66.67%
	$SeCo^{\pm}$	8.50	2.22	18.83	1.77%	1.96%	1.96%	96.08%	0.00%	33.33%	66.67%
	$SeCo^{\pm}_{opt}$	7.00	1.93	13.50	3.70%	4.76%	2.38%	92.86%	0.00%	33.33%	66.67%
YEAST	SBR^{+}_{xy}	7.43	3.28	24.36	25.51%	24.04%	27.88%	48.08%	0.00%	100.00%	0.00%
	$SBR^{\pm}_{xy}$	8.93	1.78	15.86	27.03%	31.20%	1.60%	67.20%	7.14%	92.86%	0.00%
	SBR^{+}_{y}	6.43	3.37	21.64	39.27%	48.89%	0.00%	51.11%	7.14%	85.71%	7.14%
	$SeCo^{+}$	2.86	2.75	7.86	6.36%	7.50%	10.00%	82.50%	0.00%	42.86%	57.14%
	$SeCo^{+}_{opt}$	3.36	2.43	8.14	7.89%	2.13%	17.02%	80.85%	0.00%	42.86%	57.14%
	$SeCo^{\pm}$	4.71	2.09	9.86	1.45%	0.00%	3.03%	96.97%	0.00%	14.29%	85.71%
	$SeCo^{\pm}_{opt}$	2.86	1.75	5.00	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
GENBASE	SBR^{+}_{xy}	1.85	1.06	1.96	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	$SBR^{\pm}_{xy}$	2.07	1.07	2.22	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	SBR^{+}_{y}	1.78	2.81	5.00	80.00%	47.92%	0.00%	52.08%	0.00%	62.96%	37.04%
	$SeCo^{+}$	0.67	1.00	0.67	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	$SeCo^{+}_{opt}$	0.67	1.00	0.67	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	$SeCo^{\pm}$	0.85	1.09	0.93	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	$SeCo^{\pm}_{opt}$	0.96	1.12	1.07	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
MEDICAL	SBR^{+}_{xy}	2.13	1.80	3.84	9.25%	0.00%	13.54%	86.46%	0.00%	20.00%	80.00%
	$SBR^{\pm}_{xy}$	2.40	1.37	3.29	8.78%	6.48%	3.70%	89.81%	0.00%	20.00%	80.00%
	SBR^{+}_{y}	1.38	5.27	7.27	74.62%	22.58%	0.00%	77.42%	0.00%	22.22%	77.78%
	$SeCo^{+}$	1.44	1.71	2.47	4.50%	3.08%	4.62%	92.31%	0.00%	11.11%	88.89%
	$SeCo^{+}_{opt}$	1.27	1.63	2.07	3.23%	1.75%	3.51%	94.74%	2.22%	4.44%	93.33%
	$SeCo^{\pm}$	1.04	1.77	1.84	3.61%	2.13%	4.26%	93.62%	2.22%	4.44%	93.33%
	$SeCo^{\pm}_{opt}$	1.00	1.44	1.44	3.08%	0.00%	4.44%	95.56%	0.00%	4.44%	95.56%
ENRON	SBR^{+}_{xy}	3.28	3.57	11.74	20.42%	8.62%	32.76%	58.62%	1.89%	56.60%	41.51%
	$SBR^{\pm}_{xy}$	3.72	1.81	6.72	26.12%	19.29%	6.60%	74.11%	3.77%	52.83%	43.40%
	SBR^{+}_{y}	2.34	3.89	9.09	41.29%	37.90%	0.00%	62.10%	11.32%	39.62%	49.06%
	$SeCo^{+}$	6.42	3.24	20.75	4.18%	0.59%	12.65%	86.76%	0.00%	41.51%	58.49%
	$SeCo^{+}_{opt}$	6.62	2.98	19.72	4.78%	1.71%	12.25%	86.04%	0.00%	52.83%	47.17%
	$SeCo^{\pm}$	1.83	2.20	4.02	2.35%	0.00%	5.15%	94.85%	0.00%	7.55%	92.45%
	$SeCo^{\pm}_{opt}$	0.91	1.63	1.47	1.28%	0.00%	2.08%	97.92%	0.00%	1.89%	98.11%
CAL500	SBR^{+}_{xy}	1.78	2.06	3.66	45.84%	37.22%	33.01%	29.77%	19.54%	58.05%	22.41%
	$SBR^{\pm}_{xy}$	2.86	1.27	3.64	43.44%	44.58%	5.02%	50.40%	20.11%	62.07%	17.82%
	SBR^{+}_{y}	2.59	2.45	6.36	81.93%	80.49%	2.22%	17.29%	54.02%	35.06%	10.92%
	$SeCo^{+}$	4.49	2.98	13.38	9.06%	4.48%	21.64%	73.88%	0.57%	48.85%	50.57%
	$SeCo^{+}_{opt}$	4.98	2.90	14.44	11.06%	6.00%	25.29%	68.71%	0.57%	56.90%	42.53%
	$SeCo^{\pm}$	0.19	1.30	0.25	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
	$SeCo^{\pm}_{opt}$	0.07	1.15	0.09	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
MEDIAMILL	SBR^{+}_{xy}	33.60	6.14	206.29	12.89%	2.42%	36.89%	60.70%	0.00%	100.00%	0.00%
	$SBR^{\pm}_{xy}$	11.14	2.86	31.87	13.51%	16.18%	10.58%	73.24%	0.00%	98.02%	1.98%
	SBR^{+}_{y}	22.66	6.16	139.55	12.78%	16.56%	0.00%	83.44%	0.99%	88.12%	10.89%
COREL16K	SBR^{+}_{xy}	6.77	2.57	17.43	45.11%	16.51%	46.14%	37.36%	4.58%	88.89%	6.54%
	$SBR^{\pm}_{xy}$	2.71	4.71	12.78	22.05%	51.81%	6.02%	42.17%	21.57%	64.05%	14.38%
	SBR^{+}_{y}	4.25	3.13	13.31	58.94%	39.69%	2.62%	57.69%	17.65%	58.82%	23.53%
BIBTEX	SBR^{+}_{xy}	9.99	2.98	29.75	9.09%	2.89%	18.82%	78.29%	0.00%	83.65%	16.35%
	$SBR^{\pm}_{xy}$	3.33	1.83	6.09	11.16%	7.18%	10.78%	82.04%	1.26%	43.40%	55.35%
	SBR^{+}_{y}	6.65	3.25	21.64	30.35%	26.28%	0.47%	73.25%	1.89%	75.47%	22.64%
COREL5K	SBR^{+}_{xy}	2.98	2.24	6.68	41.14%	20.65%	36.98%	42.37%	5.08%	57.75%	37.17%
	$SBR^{\pm}_{xy}$	1.61	2.59	4.17	35.43%	50.17%	9.14%	40.70%	23.26%	44.12%	32.62%
	SBR^{+}_{y}	2.12	2.49	5.28	52.15%	42.80%	2.53%	54.67%	13.90%	30.48%	55.61%

approach	EMOTIONS	SCENE	YEAST	GENBASE	MEDICAL	ENRON	CAL500	MEDIAMILL	COREL16K	BIBTEX	COREL5K
SBR^{+}_{xy}	42	73	104	49	95	173	309	3393	1035	1588	1114
$SBR^{\pm}_{xy}$	39	94	125	55	108	197	497	1125	414	529	602
SBR^{+}_{y}	31	49	90	48	62	124	450	2288	650	1057	792
PCR	197	784	724	48	304	553	167	1000	1000	1000	1000

Table 7 Total number of rules of the SBR and the PCR approaches. Note that the maximum number of rules of PCR is 1000.

To illustrate this effect, imagine a hierarchical structure on the labels. A typical rule encountered which is certainly true would be *parent*←*child* for some pair of parent and child label. In fact, it would not be surprising if it was possible to define a parent label only by rules depending on the children labels, from which the stacked approach would particularly benefit. In the other direction, it is not possible to predict the child label only based on the presence of the parent label, i.e., without additional information. Moreover, whereas the stacking approach may reflect a strong pairwise relation between two labels twice, i.e., in both label models, SeCo is tailored towards only finding one direction of the relation, preferably the one where the "easier" label is in the body. It may even appear that none of such induction chains appear due to the particular order and covering of the found rules although other approaches find relationships, as for $SeCo^{\pm}_{opt}$ on YEAST. In such cases, the algorithm may still find a good solution.

For (7) it is also remarkable that pruning substantially increases the percentage of used label features (especially for the mixed head variants). Pruning tries to remove conditions and rules which work good on a training set, but do not generalize well on a separate validation set. Hence, this increase indicates that label features are more useful for obtaining models that generalize better than the original instance features. For the iterative separate-and-conquer approach, post-optimization seems to help in order to reduce the size of the models, but unfortunately the effect on the label conditions is not clear in contrast to usual pruning.

6.3.2 Model Sizes

For the smaller datasets pruning on the positive head models decreases the number of used conditions for SBR^{+} substantially so that in the end the models use less conditions than BR^{+} for expressing the same concepts. This comes hand in hand with a decrease in the size of the positive head models directly comparing BR and stacked BR, as can be seen from the average size of the label rulesets (columns (3) and (6)). For a higher number of labels the input feature space also increases. Therefore, it becomes more likely that a higher number of features is used in order to express a rule. For example, stacking increases the input space from 68 to 241 dimensions for CAL500.

Interestingly, for the mixed head models, $SBR^{\pm}$ produces more complex models than $BR^{\pm}$ even on the smallest dataset (pruned, as for the remainder of the analysis). Still, the $\pm$ models are consistently more compact than the one-sided $+$ models (columns (3) and (6), except for BR on CAL500). This is somehow contradictory to the intuition in rule learning that it should be much easier to cover the smaller class since it contains less examples. However, as we will see further on, $BR^{\pm}$ and $SBR^{\pm}$ are also almost always worse w.r.t. prediction quality than their counterparts producing positive heads.

This is not true for the SeCo approaches, for which using positive and negative rules reduces the size of the models but usually also outperforms the positive rules (except for the special case of CAL500). The iterative separate-and-conquer approach also results in clearly smaller model sizes for the mixed heads variant compared to the combined statistics of SBR. For the positive head variants, this is only observable for the smaller datasets.

In Table 7 we include a comparison to the number of rules induced by the predictive clustering rules approach. Unfortunately, PCR in the default settings stops at the 1000th rule. However, we can still observe that the stacking approaches (including the rules from the BR models) induce a lower number of rules with the only clear exception of CAL500. As we will see further below, PCR does not benefit from the increased number of rules regarding the classification performance.

Approach	YEAST	MEDICAL	ENRON
BR$^+$	*Class4* ← x23 > 0.08, x49 < -0.09 *Class4* ← x68 < 0.05, x33 > 0.00, x24 > 0.00, x66 > 0.00, x88 > -0.06 *Class4* ← x3 < -0.03, x71 > 0.03, x91 > -0.01 *Class4* ← x68 < 0.03, x83 > -0.00, x44> 0.029, x93 < 0.01 *Class4* ← x96 < -0.03, x10 > 0.01, x78< -0.07	*Cough* ← "cough", $\overline{\text{"lobe"}}$ *Cough* ← "cough", "atelectasis" *Cough* ← "cough", opacity *Cough* ← "cough", airways *Cough* ← "cough" , $\overline{\text{"pneumonia"}}$, $\overline{\text{"2"}}$ *Cough* ← "coughing" *Cough* ← "cough", "early"	*Joke* ← "mail", "fw", "didn"
SBR$^+$	*Class4* ← *Class3*, $\overline{\textit{Class2}}$ *Class4* ← *Class5*, $\overline{\textit{Class6}}$ *Class4* ← *Class3*, *Class1*, x22 > -0.02	*Cough* ← "cough" , $\overline{\textit{Pneumonia}}$, $\overline{\textit{Pulmonary\_collapse}}$, $\overline{\textit{Asthma}}$ *Cough* ← "coughing" *Cough* ← *Asthma*, "mild"	*Joke* ← *Personal*, "day", "mail"
SeCo$^+_{opt}$	*Class4* ← *Class3*, x91 > -0.02, x50 < -0.02, x68 < 0.03 *Class4* ← *Class3*, x90 > -0.02, x77 < -0.04 *Class4* ← x60 < -0.03, x57 < -0.07, x19 > -0.01	*Cough* ← "cough", "lobe", $\overline{\text{"asthma"}}$ *Cough* ← "cough", "opacity" *Cough* ← "cough", "atelectasis" *Cough* ← "cough", "airways" *Cough* ← "cough", *Fever*	*Joke* ← "didn", $\overline{\text{"wednesday"}}$ *Joke* ← *Personal*, "forwarded"

Fig. 8 Example rulesets for one exemplary label, respectively, learned by BR, SBR and SeCo. Attribute names in italic denote label attributes, attributes with an overline denote negated conditions.

6.3.3 Dependencies

Whereas (7) may serve as an indicator of general dependency between labels, columns (8-10) and especially (11-13) in Table 5 allow to further differentiate. For instance, the percentage of fully label-dependent rulesets in the model of SBR$^+$ for YEAST shows that 14.29% of the labels can be induced only based on other labels regardless of any instance feature. This is a strong indicator for a high degree of *unconditional* label dependencies in the dataset. If we additionally consider the absence of labels ($\pm$), we obtain in the stacked model a rate of 57.14% of labels which are unconditionally dependent on other labels. For one of the labels the learned label-dependent rule is even the only rule which can actually produce a prediction except for the default rule (see column (11) in Table 6)[4] From the number of partial rulesets we can estimate a bound on the number of conditional dependencies in the data. However, note that (11) as well as (13) substantially suffers from a kind of *feature flooding* noise: The probability of selecting an instance or label feature in the refinement step of a rule instead of an equally good feature

[4] The rule improves the precision for that label from 74% to 99% at a recall of 100% on the training data.

or label feature, respectively, increases with growing number of total instance and label features. This can also be seen to a certain degree for SBR$^{\pm}$ on the EMOTIONS dataset, where the high number of full label-dependent rules does not lead to any pure label-dependent ruleset.

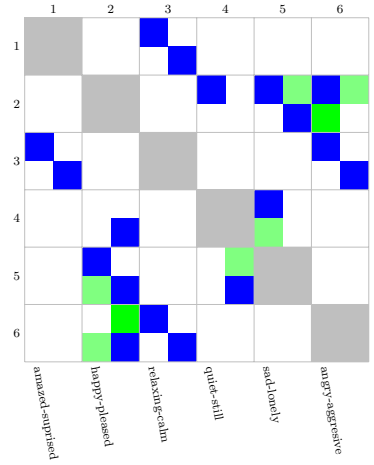
The datasets with the highest observed degree of label dependency are YEAST for the small ones and CAL500, COREL5K and COREL16K for the larger ones. For CAL500, this may be explained by the categorizations of songs into emotions, which often come hand in hand or completely contradict, like *Angry-Agressive* against *Carefree-Lighthearted*.

This can also be observed from Figures 9 and 10. The figures presents graphically the label-dependent rules found by $SBR^{\pm}$ on the different datasets except GENBASE. Labels are enumerated from 1 to the number of labels and the corresponding label names are visible on the bottom of the coordinate system for the smaller datasets. Blue boxes in the intersection square between a *row label* and a *column label* depict fully label-dependent rules, green boxes show partially label-dependent rules. A colored box at the top corners indicates a rule of the type *row label*$\leftarrow \ldots$,*column label*,..., whereas the bottom corners represent the opposite *column label*$\leftarrow \ldots$,*row label*,... rules. The intensity of the color depends on the number of found rules normalized by the maximum number of rules found for any pairwise label combination. Only blue boxes in an intersection square indicate that only fully label-dependent rules were found between the respective two labels.

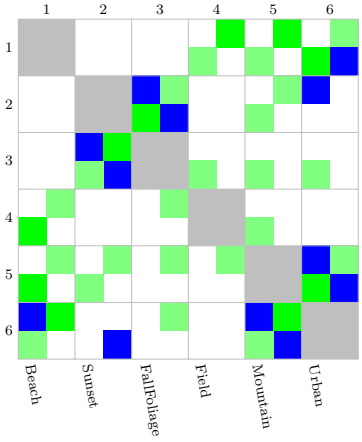
As an example, the blue box in the upper left corner of the square in the second row and fourth column in Figure 9a indicates that the algorithm found a rule of the type *happy-pleased*$\leftarrow$,...*quiet-still*,.... However, it is not possible to find out from the picture whether the head or conditions are positive or negative. The EMOTIONS dataset is concerned with assigning moods to audio samples.

We can observe for SCENE that dependencies often also depend on some instance features. This is reasonable, since the task in this dataset is to predict elements of a scenery image, and although some label combinations may be more likely than others, whether an element is present or not will still most certainly depend on the content of the picture at hand. In YEAST the labels seem to be organized in a special way since we encounter the pattern that a label depends on its preceding and the two succeeding labels. ENRON has a hierarchical structure on its labels, which can be recognized from the vertical and horizontal patterns originating from parent labels. The different clusters in CAL500 are due to the dataset containing labels of different, orthogonal categories, e.g., emotions as well as genre labels. The stripes results from sets of tightly coupled labels which in addition appear in succession in the original ordering in the data. For instance, the long diagonal of around 10 squares on the bottom or right, respectively, originates from the connection between the genres *rock – best rock*, *r'n'b – best r'n'b*, *pop – best pop*, etc.

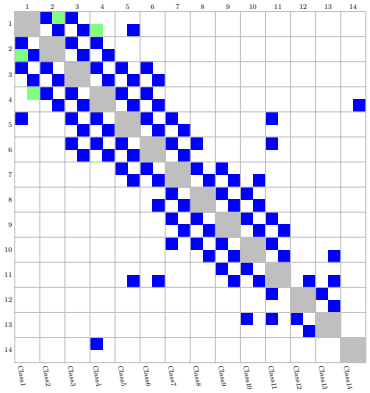
The datasets in Figure 10 are more difficult to inspect due to the high number of labels, however, we can also observe some regularities. Datasets COREL16K and COREL5K from image segment recognition are concerned with the tagging of images. Long orthogonal stripes correspond to labels such as *water* (long stripe on the right/bottom of Figure 10b), *street* or *people*. Tagging videos in MEDIAMILL apparently involves many frequent and co-occurring labels. Unfortunately the labels do not have any meaningful names. Figure 10c reveals relations between tags such as *semantic* and *web* or *knowledge* and *management* in the social bookmarking system of BibSonomy.



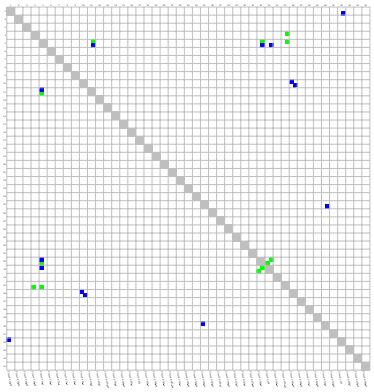
(a) EMOTIONS



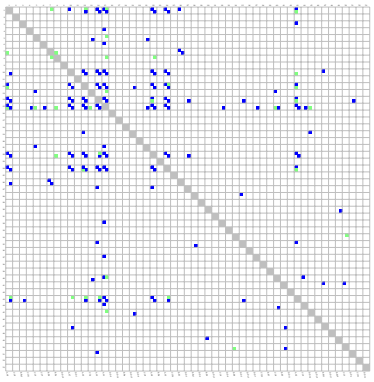
(b) SCENE



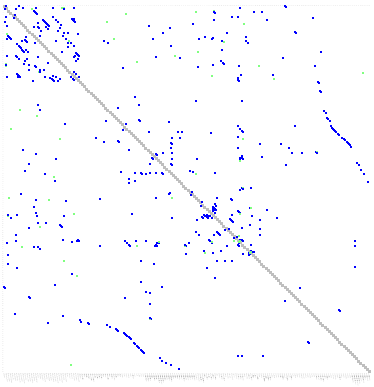
(c) YEAST



(d) MEDICAL

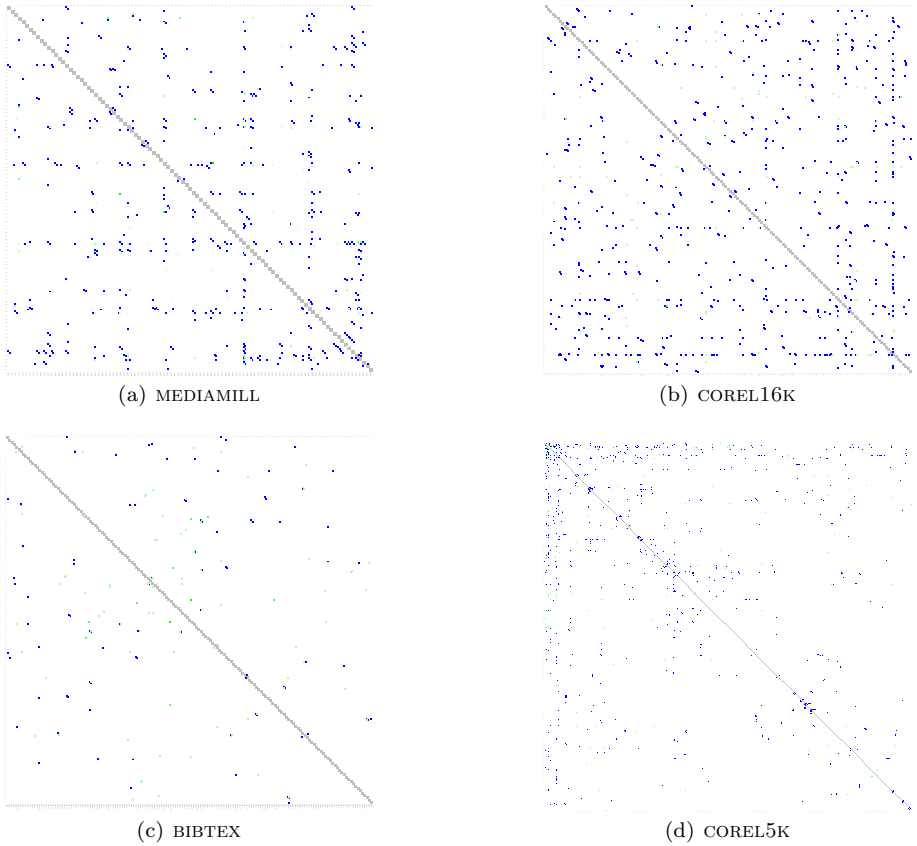


(e) ENRON



(f) CAL500

Fig. 9 Graphical representation of the found label-dependent rules for SBR$^{\pm}$ on all datasets except GENBASE, first part.



(a) MEDIAMILL (b) COREL16K

(c) BIBTEX (d) COREL5K

Fig. 10 Graphical representation of the found label-dependent rules for SBR$^{\pm}$ on all datasets except GENBASE, second part.

6.3.4 Exemplary Rule Models

Examples of learned rulesets for YEAST are given in Figure 8. In this particular case, we see a much more compact and less complex ruleset for *Class4* for the stacked model than for the independently learned BR classifier. The ruleset also seems more appropriate for a domain expert to understand coherences between proteins (instance features) and protein functions (labels). The SeCo model is less explicit in this sense, but it shows that certainly *Class3* is an important class for expressing *Class4*.[5] Remind that SeCo^{+} with positive heads is not able to express dependencies on absent labels since this information is never added to the training set.

Figure 8 also shows the models for the diagnosis *Cough* in the MEDICAL task. This dataset is concerned with the assignment of international diseases codes (ICD)

[5] For convenience, we only show the rules with this label in the head.

to real, free text radiological reports.[6] Interestingly, the stacked model reads very well, and the found relationship seems to be even comprehensible by non-experts: If the patient does not have *Pneumonia*, a *Pulmonary_collapse* or *Asthma* and "cough"s or is "coughing", he just has a *Cough*. Otherwise, he may also have a "mild" *Asthma*, in which case he is also considered to have a *Cough*. For SeCo, *Cough* only depends on label *Fever* at the end. But note that from SBR's model *Cough* seems to be opposing to *Pneumonia* and *Pulmonary_collapse*, a relationship which cannot be modeled by SeCo$^+$. SeCo$^\pm$ in contrast is able to find a rule *Pneumonia* $\leftarrow \overline{Cough}$, "lobe".

In ENRON, which is concerned with the categorization of emails during the Enron scandal, the model is less comprehensible, as it is also for the BR model. However, the relation between *Personal* and *Joke* can clearly be explained from the hierarchical structure on the topics. This also shows the potential of using rule learning, particularly with the stacking approach, in MLC for reconstructing underlying hierarchies.

6.4 Predictive Performance

Until now, our analysis mainly focused on the inspection of the multi-label rules as obtaining these is the main objective of this work. However, if the predictive performance of the rulesets was far below compared to state-of-the-art methods, the implications made based on the model analysis would also be questionable. We will show in this section that the induced rules can indeed be used effectively in order to produce accurate predictions.

Note that the different models shown in the previous sections can be used in different ways in order to produce predictions. More specifically, we consider SBR with abstaining (SBR$_?$) and predicting the default label (SBR$_d$) separately. The results are shown in Tables 8 and 9 together with the performances of BR and SeCo and in Table 10 excepting SeCo. Table 10 presents the result on the larger datasets, on which SeCo did not finish. Since the outcomes on these datasets revealed more extreme values, they are discussed separately in Section 6.4.1. Additional variants such as learning only from the label features (SBR$_y$), Gibbs sampling (SBR$_{Gibbs}$) and the predictive clustering rules classifier (PCR) are shown in separate blocks (Section 6.4.1). For the sake of clarity, we only report the results after the 10^{th} bootstrapping iteration in the case of SBR.[7] All the different results for SBR were obtained from only two models SBR$^+$ and SBR$^\pm$. Section 6.4.2 reveals statistically significant differences.

Our first observation is that predicting only the presence of labels is almost always better for BR on all measures on the smaller dataset except for some cases, where the differences are small. This is remarkable, since the picture is not that clear after the predictions of these classifiers are passed through the stacked models, especially if we consider recall and precision separately. SBR$^+_?$ is the best overall approach in terms of micro F1 on the first seven datasets, but for micro precision, e.g., it is outperformed by its $\pm$ counterpart on four of the datasets of

[6] We generated a special version of this dataset directly from the sources [38] for this figure, since the label names in the publicly available dataset are incorrect.

[7] We found that increasing number of iterations consistently decrease recall, but increases precision and F1. However, the average absolute difference was consistently below 1%.

Table 8 Experimental performance of all the approaches on the first seven datasets (part 1). The methods are divided into two blocks. The small number after the metric indicates the rank of the particular approach w.r.t the dataset.

Approach	Hamming Acc.	Subset Acc.	Mi. Precision	Mi. Recall	Mi. F1	Ma. Precision	Ma. Recall	Ma. F1
	EMOTIONS							
BR^{+}	77.21% 5	23.60% 7	65.54% 2	57.23% 12	60.97% 9	65.61% 1	55.80% 10	57.54% 9
$BR^{\pm}$	76.08% 7	19.22% 11	62.53% 7	58.16% 10	60.03% 10	62.74% 4	55.75% 11	55.17% 11
$SBR^{+}_{?}$	77.32% 4	24.28% 6	64.02% 5	62.73% 7	63.24% 4	62.70% 5	61.30% 6	59.82% 5
$SBR^{\pm}_{?}$	77.20% 6	24.45% 5	63.53% 6	64.59% 5	63.78% 3	60.52% 7	62.80% 5	59.79% 6
SBR^{+}_{d}	75.02% 9	24.96% 2	57.46% 9	75.54% 4	65.24% 1	57.99% 9	74.35% 4	63.96% 1
$SBR^{\pm}_{d}$	75.44% 8	24.95% 3	60.39% 8	63.15% 6	61.55% 8	63.45% 3	60.43% 7	59.29% 7
$SeCo^{+}$	63.92% 14	4.89% 14	46.09% 14	83.96% 1	59.30% 12	50.70% 14	83.11% 1	60.43% 3
$SeCo^{+}_{opt}$	66.20% 13	5.88% 13	47.78% 13	78.87% 3	59.33% 11	51.97% 13	78.25% 3	60.14% 4
$SeCo^{\pm}$	77.48% 2	22.91% 9	65.23% 4	59.27% 8	61.93% 6	60.63% 6	56.12% 9	56.21% 10
$SeCo^{\pm}_{opt}$	78.61% 1	26.63% 1	68.80% 1	57.30% 11	62.42% 5	58.28% 8	54.06% 13	54.52% 12
$SBR^{+}_{y/?}$	77.40% 3	24.62% 4	65.41% 3	58.93% 9	61.90% 7	65.08% 2	57.41% 8	58.51% 8
$SBR^{+}_{y/d}$	72.91% 10	23.45% 8	54.47% 11	79.46% 2	64.59% 2	54.84% 11	78.28% 2	63.52% 2
SBR^{+}_{Gibbs}	70.38% 12	18.03% 12	52.82% 12	54.74% 14	53.46% 14	53.23% 12	52.29% 14	49.12% 14
PCR	72.53% 11	20.39% 10	55.96% 10	55.77% 13	55.80% 13	55.26% 10	54.73% 12	54.13% 13
	SCENE							
BR^{+}	88.03% 3.5	46.24% 5	68.82% 3.5	60.94% 7.5	64.55% 3.5	69.01% 4.5	62.40% 7.5	64.95% 3.5
$BR^{\pm}$	86.83% 7	41.72% 11	65.43% 7	58.88% 11	61.55% 9	69.96% 3	59.32% 11	62.44% 10
$SBR^{+}_{?}$	87.66% 5	46.11% 7	65.56% 6	65.68% 5	65.58% 2	65.73% 7	67.36% 5	65.72% 2
$SBR^{\pm}_{?}$	88.20% 1	46.74% 3	70.37% 2	59.33% 10	64.27% 5	71.00% 2	60.69% 10	64.65% 5
SBR^{+}_{d}	86.04% 9	45.32% 10	58.31% 10	77.79% 4	66.63% 1	60.07% 10	78.80% 4	67.82% 1
$SBR^{\pm}_{d}$	86.41% 8	48.40% 1	61.57% 8	64.48% 6	62.93% 7	64.85% 8	65.26% 6	64.39% 6
$SeCo^{+}$	78.15% 12	24.55% 13	44.47% 12	81.83% 2	57.46% 11	55.23% 12	81.64% 2	63.29% 9
$SeCo^{+}_{opt}$	79.73% 11	23.97% 14	46.41% 11	80.24% 3	58.73% 10	56.39% 11	79.97% 3	63.91% 7
$SeCo^{\pm}$	87.58% 6	45.36% 9	67.39% 5	59.71% 9	63.12% 6	68.78% 6	61.41% 9	63.71% 8
$SeCo^{\pm}_{opt}$	88.06% 2	45.82% 8	71.71% 1	54.95% 12	62.16% 8	71.89% 1	56.58% 12	62.25% 11
$SBR^{+}_{y/?}$	88.03% 3.5	46.24% 5	68.82% 3.5	60.94% 7.5	64.55% 3.5	69.01% 4.5	62.40% 7.5	64.95% 3.5
$SBR^{+}_{y/d}$	73.34% 14	46.24% 5	38.78% 13	83.14% 1	52.85% 13	38.69% 13	84.04% 1	52.64% 13
SBR^{+}_{Gibbs}	85.18% 10	48.15% 2	59.61% 9	53.80% 13	56.53% 12	60.56% 9	54.14% 13	54.86% 12
PCR	77.88% 13	33.90% 12	37.97% 14	37.13% 14	37.55% 14	38.48% 14	37.58% 14	37.72% 14
	YEAST							
BR^{+}	78.77% 1	9.18% 6	68.47% 2	55.33% 8	61.19% 5	47.45% 1	32.48% 10	34.91% 7
$BR^{\pm}$	68.11% 12	1.24% 11	46.42% 12	39.74% 13	42.25% 13	34.53% 8	34.11% 7	28.89% 11
$SBR^{+}_{?}$	78.53% 4	10.18% 3	66.88% 4	57.63% 6	61.90% 2	47.25% 2	34.82% 6	36.83% 5
$SBR^{\pm}_{?}$	73.43% 8	2.15% 10	55.83% 7	62.37% 5	58.71% 7	35.57% 7	39.17% 5	33.54% 9
SBR^{+}_{d}	75.41% 6	10.18% 4	58.31% 6	66.21% 4	61.98% 1	39.84% 4	43.53% 4	40.01% 4
$SBR^{\pm}_{d}$	72.84% 10	5.34% 8	55.62% 9	49.51% 12	52.03% 10	28.55% 14	30.30% 11	28.34% 12
$SeCo^{+}$	41.77% 13	0.00% 13.5	34.03% 13	98.18% 1	50.53% 11	30.42% 11	87.95% 1	42.02% 1
$SeCo^{+}_{opt}$	41.43% 14	0.00% 13.5	33.84% 14	97.71% 2	50.25% 12	30.42% 12	87.59% 2	41.80% 2
$SeCo^{\pm}$	68.66% 11	0.95% 12	48.40% 11	31.71% 14	37.90% 14	31.35% 10	27.04% 14	24.25% 13
$SeCo^{\pm}_{opt}$	78.66% 2	8.85% 7	69.09% 1	53.41% 9	60.23% 6	33.44% 9	28.95% 12	29.31% 10
$SBR^{+}_{y/?}$	78.61% 3	9.47% 5	67.28% 3	57.10% 7	61.76% 3	46.97% 3	34.07% 8	36.06% 6
$SBR^{+}_{y/d}$	73.71% 7	10.80% 2	55.22% 10	69.97% 3	61.70% 4	38.75% 5	46.64% 3	40.08% 3
SBR^{+}_{Gibbs}	75.94% 5	4.88% 9	62.28% 5	51.99% 10	56.64% 8	29.75% 13	27.05% 13	23.92% 14
PCR	72.88% 9	10.88% 1	55.62% 8	51.77% 11	53.60% 9	37.07% 6	33.90% 9	34.85% 8
	GENBASE							
BR^{+}	99.88% 2	96.83% 2	98.95% 8	98.42% 3.5	98.68% 2	95.19% 2	95.67% 2	95.33% 2
$BR^{\pm}$	99.03% 12	83.24% 9	99.38% 4	80.11% 12	88.54% 12	72.21% 12	72.57% 11.5	72.36% 12
$SBR^{+}_{?}$	99.88% 2	96.83% 2	98.95% 8	98.42% 3.5	98.68% 2	95.19% 2	95.67% 2	95.33% 2
$SBR^{\pm}_{?}$	99.04% 11	82.94% 11.5	99.55% 1	80.27% 11	88.68% 11	72.28% 11	72.57% 11.5	72.40% 11
SBR^{+}_{d}	99.88% 2	96.83% 2	98.95% 8	98.42% 3.5	98.68% 2	95.19% 2	95.67% 2	95.33% 2
$SBR^{\pm}_{d}$	99.02% 13	82.94% 11.5	99.55% 2	79.66% 13	88.31% 13	71.17% 14	71.46% 14	71.29% 14
$SeCo^{+}$	99.18% 9.5	79.30% 13.5	87.19% 13.5	96.69% 6.5	91.67% 9.5	91.04% 9.5	92.31% 8.5	91.24% 9.5
$SeCo^{+}_{opt}$	99.18% 9.5	79.30% 13.5	87.19% 13.5	96.69% 6.5	91.67% 9.5	91.04% 9.5	92.31% 8.5	91.24% 9.5
$SeCo^{\pm}$	99.72% 6	92.76% 8	99.26% 6	94.82% 10	96.98% 6	92.46% 6	92.51% 7	92.35% 7
$SeCo^{\pm}_{opt}$	99.76% 5	93.51% 7	99.29% 5	95.60% 9	97.39% 5	94.09% 4	93.70% 6	93.71% 5
$SBR^{+}_{y/?}$	99.80% 4	95.32% 5.5	97.38% 10	98.42% 3.5	97.88% 4	93.37% 5	94.93% 5	93.86% 4
$SBR^{+}_{y/d}$	99.64% 8	95.32% 5.5	94.45% 12	98.67% 1	96.38% 8	91.59% 8	94.95% 4	92.79% 6
SBR^{+}_{Gibbs}	99.01% 14	83.09% 10	99.53% 3	79.51% 14	88.25% 14	71.56% 13	71.83% 13	71.67% 13
PCR	99.66% 7	95.77% 4	96.70% 11	96.16% 8	96.40% 7	92.25% 7	92.15% 10	91.78% 8

the smaller and on two of the bigger datasets. Also surprisingly, it seems to be better for $SBR^{\pm}$ to abstain from classification if the prediction is negative even though this approach explicitly also learns and applies rules with negative heads. Regarding the relatively bad performance of the base $BR^{\pm}$, this might indicate that the mixed heads version of Ripper has an over-generalization problem when learning negative heads.

Table 9 Experimental performance of all the approaches on the first seven datasets (part 2). The last block shows the average over the ranks on the first seven datasets.

Approach	Hamming Acc.	Subset Acc.	Mi. Precision	Mi. Recall	Mi. F1	Ma. Precision	Ma. Recall	Ma. F1
	MEDICAL							
BR^{+}	98.99% 3	66.96% 3	80.26% 7	84.29% 7	82.19% 1	76.57% 1	79.39% 3	77.39% 1
$BR^{\pm}$	98.51% 9	48.26% 11	84.61% 1	56.38% 13	67.58% 11	59.90% 13	59.67% 14	59.54% 14
$SBR^{+}_{?}$	98.97% 4	66.86% 4	79.38% 8	84.78% 5	81.96% 3	75.28% 5	78.36% 6	76.21% 3
$SBR^{\pm}_{?}$	98.55% 8	51.22% 9	84.23% 3	58.97% 11	69.31% 9	60.21% 11	61.14% 12	60.45% 12
SBR^{+}_{d}	98.95% 5	66.25% 5	78.21% 9	86.01% 2	81.89% 4	75.02% 6	78.51% 5	76.10% 4
$SBR^{\pm}_{d}$	98.58% 7	54.09% 8	82.10% 5	62.79% 10	71.09% 7	59.83% 14	61.77% 10	60.54% 11
$SeCo^{+}$	96.96% 14	19.23% 13	48.11% 14	85.69% 4	61.38% 13	74.09% 8	77.77% 7	74.24% 8
$SeCo^{+}_{opt}$	97.29% 13	10.94% 14	50.59% 13	87.76% 1	64.16% 12	74.42% 7	79.79% 1	75.47% 6
$SeCo^{\pm}$	99.02% 2	68.20% 1	84.14% 4	79.73% 9	81.81% 5	75.77% 4	76.67% 8	75.55% 5
$SeCo^{\pm}_{opt}$	99.03% 1	67.99% 2	84.25% 2	80.13% 8	82.09% 2	75.79% 3	76.40% 9	75.47% 7
$SBR^{+}_{y/?}$	98.90% 6	64.51% 6	77.72% 10	84.37% 6	80.87% 6	76.24% 2	79.40% 2	77.20% 2
$SBR^{+}_{y/d}$	97.93% 11	62.36% 7	58.91% 11	85.94% 3	69.74% 8	69.71% 9	79.39% 4	72.44% 9
SBR^{+}_{Gibbs}	98.46% 10	50.10% 10	80.45% 6	58.65% 12	67.73% 10	60.12% 12	60.45% 13	59.94% 13
PCR	97.44% 12	35.79% 12	54.43% 12	46.52% 14	50.12% 14	64.42% 10	61.38% 11	61.96% 10
	ENRON							
BR^{+}	94.90% 2	9.17% 3	62.75% 3	49.09% 9	55.03% 3	34.20% 1	28.53% 9	29.59% 3
$BR^{\pm}$	94.66% 5	6.52% 10	61.48% 4	43.88% 11	51.16% 5	31.71% 4	28.41% 10	28.58% 5
$SBR^{+}_{?}$	94.58% 6	9.17% 4	57.96% 6	55.09% 5	56.40% 2	32.74% 3	30.34% 5	30.15% 1
$SBR^{\pm}_{?}$	93.75% 8	1.76% 11	51.19% 8	50.99% 8	50.94% 7	28.76% 8	28.88% 8	27.83% 8
SBR^{+}_{d}	92.33% 11	9.87% 2	43.13% 11	59.06% 3	49.71% 8	29.19% 7	32.85% 4	29.05% 4
$SBR^{\pm}_{d}$	92.66% 10	6.82% 9	44.31% 10	53.75% 6	48.37% 9	28.03% 10	29.89% 6	27.59% 10
$SeCo^{+}$	80.33% 14	0.00% 14	22.25% 14	82.83% 1	35.03% 14	23.79% 14	43.72% 2	26.69% 12
$SeCo^{+}_{opt}$	81.50% 13	0.12% 13	23.27% 13	82.02% 2	36.21% 13	25.13% 13	43.75% 1	27.75% 9
$SeCo^{\pm}$	94.79% 3	9.05% 5	63.60% 2	43.10% 12	51.07% 6	30.96% 5	27.80% 11	28.13% 6
$SeCo^{\pm}_{opt}$	94.99% 1	10.05% 1	65.83% 1	44.47% 10	53.03% 4	28.74% 9	26.69% 12	27.06% 11
$SBR^{+}_{y/?}$	94.78% 4	8.70% 6	60.41% 5	53.05% 7	56.41% 1	34.15% 2	29.28% 7	29.81% 2
$SBR^{+}_{y/d}$	89.97% 12	8.11% 7	33.05% 12	55.71% 4	41.45% 11	27.89% 11	34.28% 3	28.08% 7
SBR^{+}_{Gibbs}	94.12% 7	1.06% 12	57.82% 7	29.02% 14	38.29% 12	27.06% 12	24.71% 14	24.71% 14
PCR	93.66% 9	7.93% 8	50.43% 9	37.35% 13	42.90% 10	29.83% 6	25.29% 13	26.35% 13
	CAL500							
BR^{+}	85.39% 4	0.00% 7.5	52.73% 4	24.88% 10	33.76% 10	22.48% 7	19.29% 11	19.79% 10
$BR^{\pm}$	83.93% 9	0.00% 7.5	44.57% 9	28.88% 8	34.84% 8	20.73% 8	22.46% 7	20.28% 9
$SBR^{+}_{?}$	84.54% 7	0.00% 7.5	47.61% 7	30.90% 5	37.42% 4	24.88% 4	22.03% 8	22.26% 7
$SBR^{\pm}_{?}$	71.72% 12	0.00% 7.5	30.09% 12	67.11% 3	41.52% 1	23.99% 6	45.42% 3	28.19% 1
SBR^{+}_{d}	83.95% 8	0.00% 7.5	44.76% 8	30.43% 7	36.20% 5	25.26% 3	22.78% 5	22.91% 6
$SBR^{\pm}_{d}$	85.71% 3	0.00% 7.5	56.44% 3	20.38% 14	29.89% 14	19.09% 12	17.77% 13	17.52% 12
$SeCo^{+}$	65.99% 14	0.00% 7.5	27.70% 13	78.75% 1	40.96% 2	19.96% 10	47.93% 1	25.23% 2
$SeCo^{+}_{opt}$	66.44% 13	0.00% 7.5	27.48% 14	75.49% 2	40.25% 3	19.15% 11	45.73% 2	24.23% 3
$SeCo^{\pm}$	85.99% 2	0.00% 7.5	58.51% 2	22.36% 12	32.25% 11	16.79% 13	17.97% 12	17.04% 13
$SeCo^{\pm}_{opt}$	86.21% 1	0.00% 7.5	61.14% 1	21.55% 13	31.84% 12	16.28% 14	17.36% 14	16.58% 14
$SBR^{+}_{y/?}$	84.99% 5	0.00% 7.5	49.93% 5	27.91% 9	35.72% 7	24.00% 5	20.75% 9	21.13% 8
$SBR^{+}_{y/d}$	83.79% 10	0.00% 7.5	44.34% 10	30.71% 6	36.15% 6	26.10% 1	23.04% 4	23.26% 4
$SBR^{\pm}_{Gibbs}$	84.81% 6	0.00% 7.5	48.62% 6	23.76% 11	31.83% 13	20.30% 9	19.76% 10	18.72% 11
PCR	82.52% 11	0.00% 7.5	39.33% 11	31.05% 4	34.67% 9	25.45% 2	22.77% 6	23.22% 5
	Average ranks							
BR^{+}	2.93 2	4.79 3	4.21 2	8.14 8	4.79 4	2.50 1	7.50 7	5.07 4
$BR^{\pm}$	8.71 9	10.07 12	6.29 6.5	11.14 13	9.71 9.5	7.43 7.5	10.21 11	10.29 12.5
$SBR^{+}_{?}$	4.57 4.5	4.79 3	6.29 6.5	5.21 5	2.71 1	4.00 3	5.43 5	3.57 2
$SBR^{\pm}_{?}$	7.71 7	8.14 10	5.57 4	7.57 7	6.14 6	7.43 7.5	7.79 8	7.43 8
SBR^{+}_{d}	7.14 6	4.64 1	8.71 10	3.93 4	3.14 2	5.86 4	4.00 4	3.14 1
$SBR^{\pm}_{d}$	8.43 8	6.86 7	6.43 8	9.57 9	9.71 9.5	10.71 11	9.57 9	10.29 12.5
$SeCo^{+}$	12.93 14	12.64 13.5	13.36 14	2.36 1	10.36 12	11.21 13	3.21 3	6.36 7
$SeCo^{+}_{opt}$	12.36 13	12.64 13.5	13.07 13	2.79 2	10.07 11	10.93 12	2.93 1	5.79 5
$SeCo^{\pm}$	4.57 4.5	7.36 8	4.86 3	10.57 11	7.71 8	7.14 6	10.00 10	8.86 9
$SeCo^{\pm}_{opt}$	1.86 1	4.79 3	1.71 1	10.29 10	6.00 5	6.86 5	11.14 13	10.00 10
$SBR^{+}_{y/?}$	4.07 3	5.57 5	5.64 5	7.00 6	4.50 3	3.36 2	6.64 6	4.79 3
$SBR^{+}_{y/d}$	10.29 11.5	6.00 6	11.29 12	2.86 3	7.43 7	8.29 10	3.00 2	6.29 6
SBR^{+}_{Gibbs}	9.14 10	8.93 11	6.86 9	12.57 14	11.86 14	11.43 14	12.86 14	13.00 14
PCR	10.29 11.5	7.79 9	10.71 11	11.00 12	10.86 13	7.86 9	10.71 12	10.14 11

Comparing BR^{+} to its extensions $SBR^{+}_{?}$ and SBR^{+}_{d}, BR^{+} is almost always better on precision, but also always worse w.r.t. recall.[8] The stacked variants however generally find the better trade-off between recall and precision, obtaining the best average rank w.r.t. F1 over all approaches. Recall again that BR's predictions are the inputs for the stacked variants. Apparently, the additional iterations applying

[8] Except for macro precision on COREL16K, macro recall and precision on CAL500 and of course for GENBASE, where all plain and stacked BR models are equal.

Table 10 Experimental performance of all the approaches except SeCo on the remaining four datasets. The last block shows the average over the ranks on all datasets (ranks excluding SeCo).

	Hamming Acc.	Subset Acc.	Mi. Precision	Mi.Recall	Mi. F1	Ma. Precision	Ma. Recall	Ma. F1
	MEDIAMILL							
BR^{+}	96.87% 1	8.50% 6	71.20% 1	46.44% 7	56.21% 3	30.94% 1	17.73% 5	20.48% 5
$BR^{\pm}$	92.41% 10	0.13% 10	31.14% 10	61.44% 1	41.28% 10	13.47% 7	15.92% 6	9.78% 6
$SBR^{+}_{?}$	96.77% 3	9.37% 3	67.28% 5	49.47% 5	57.01% 1	28.08% 3	20.76% 3	21.76% 1
$SBR^{\pm}_{?}$	94.98% 9	0.21% 9	43.83% 9	52.35% 2	47.55% 7	8.70% 8	7.66% 7	6.13% 7
SBR^{+}_{d}	96.30% 6	9.91% 1	58.46% 7	50.41% 3	54.13% 4	23.12% 5	23.70% 1	21.58% 2
$SBR^{\pm}_{d}$	96.60% 4	8.98% 5	71.05% 2	36.23% 9	47.98% 6	8.04% 9	4.44% 9	5.07% 8
$SBR^{+}_{y/?}$	96.83% 2	9.28% 4	69.48% 4	47.97% 6	56.75% 2	30.32% 2	18.93% 4	21.28% 3
$SBR^{+}_{y/d}$	96.29% 7	9.54% 2	58.49% 6	49.49% 4	53.60% 5	25.52% 4	21.49% 2	21.02% 4
SBR^{+}_{Gibbs}	95.53% 8	2.89% 8	48.71% 8	39.76% 8	43.54% 9	5.71% 10	5.58% 8	4.56% 9
PCR	96.51% 5	5.79% 7	70.57% 3	33.20% 10	45.16% 8	15.11% 6	3.05% 10	3.42% 10
	COREL16K							
BR^{+}	98.10% 2	0.51% 6	39.31% 1	2.82% 7	5.26% 7	11.91% 3	2.24% 6	3.34% 5
$BR^{\pm}$	97.97% 8	0.04% 8	32.64% 5	0.76% 9	1.17% 9	0.66% 9	0.57% 9	0.41% 9
$SBR^{+}_{?}$	97.92% 9	0.06% 7	30.81% 7	1.81% 8	2.44% 8	0.72% 8	0.67% 8	0.48% 8
$SBR^{\pm}_{?}$	98.09% 4	0.67% 4	37.60% 3	3.15% 5	5.81% 4	11.99% 2	2.53% 4	3.73% 3
SBR^{+}_{d}	98.07% 5	0.85% 3	34.41% 4	3.77% 3	6.79% 3	12.09% 1	2.99% 3	4.32% 1
$SBR^{\pm}_{d}$	98.13% 1	0.00% 10	3.33% 10	0.00% 10	0.01% 10	0.35% 10	0.33% 10	0.33% 10
$SBR^{+}_{y/?}$	98.10% 3	0.57% 5	38.46% 2	2.96% 6	5.49% 6	11.86% 4	2.32% 5	3.45% 4
$SBR^{+}_{y/d}$	97.99% 7	1.07% 1	31.32% 6	4.95% 2	8.18% 2	11.67% 5	3.04% 2	4.21% 2
SBR^{+}_{Gibbs}	95.24% 10	0.02% 9	10.41% 9	20.16% 1	13.51% 1	1.17% 7	4.10% 1	1.18% 7
PCR	98.01% 6	1.04% 2	24.82% 8	3.18% 4	5.63% 5	7.72% 6	2.03% 7	2.79% 6
	BIBTEX							
BR^{+}	98.60% 1	15.92% 3	55.48% 5	37.34% 5	44.63% 4	38.05% 1	29.58% 5	30.85% 4
$BR^{\pm}$	98.60% 3	3.58% 10	89.75% 2	8.53% 9	15.49% 9	2.18% 9	1.56% 9	1.71% 9
$SBR^{+}_{?}$	98.41% 8	15.73% 5	47.10% 7	44.65% 2	45.83% 2	36.20% 3	37.25% 2	34.35% 2
$SBR^{\pm}_{?}$	98.54% 6	4.69% 8	80.82% 3	9.55% 8	16.55% 8	2.34% 8	1.88% 8	1.91% 8
SBR^{+}_{d}	98.38% 9	15.74% 4	46.43% 8	45.38% 1	45.89% 1	35.84% 4	38.06% 1	34.56% 1
$SBR^{\pm}_{d}$	98.60% 2	3.76% 9	93.72% 1	8.01% 10	14.72% 10	1.75% 10	1.28% 10	1.39% 10
$SBR^{+}_{y/?}$	98.56% 5	16.06% 1	53.19% 6	39.19% 4	45.12% 3	37.72% 2	31.25% 4	31.59% 3
$SBR^{+}_{y/d}$	98.00% 10	15.96% 2	36.18% 10	42.42% 3	39.02% 5	34.31% 5	34.44% 3	30.34% 5
SBR^{+}_{Gibbs}	98.59% 4	4.77% 7	74.09% 4	10.58% 7	18.48% 6	4.15% 7	2.71% 7	2.87% 7
PCR	98.44% 7	8.95% 6	44.24% 9	11.38% 6	18.09% 7	14.01% 6	6.92% 6	8.32% 6
	COREL5K							
BR^{+}	99.03% 2	0.32% 6	38.31% 2	4.68% 8	8.34% 8	34.94% 2	31.59% 5	32.11% 4
$BR^{\pm}$	98.63% 10	0.06% 8	16.12% 10	10.80% 1	12.39% 1	31.27% 7	31.74% 3	31.20% 7
$SBR^{+}_{?}$	99.02% 5	0.72% 4	36.07% 3	6.00% 6	10.28% 6	34.80% 3	31.53% 6	32.10% 5
$SBR^{\pm}_{?}$	99.02% 4	0.00% 9.5	29.49% 6	1.55% 9	2.49% 9	30.94% 9	30.91% 9	30.88% 9
SBR^{+}_{d}	98.99% 7	0.84% 3	32.23% 4	6.98% 4	11.47% 4	34.38% 5	31.39% 8	31.94% 6
$SBR^{\pm}_{d}$	99.04% 1	0.00% 9.5	18.63% 8	0.57% 10	0.90% 10	30.88% 10	30.86% 10	30.85% 10
$SBR^{+}_{y/?}$	99.03% 3	0.56% 5	38.38% 1	5.41% 7	9.47% 7	35.06% 1	31.69% 4	32.25% 3
$SBR^{+}_{y/d}$	98.99% 6	0.88% 2	32.13% 5	6.79% 5	11.19% 5	34.51% 4	31.77% 2	32.27% 2
SBR^{+}_{Gibbs}	98.65% 9	0.08% 7	16.48% 9	10.12% 2	12.14% 3	31.18% 8	31.49% 7	31.11% 8
PCR	98.90% 8	0.98% 1	24.35% 7	8.17% 3	12.23% 2	33.84% 6	32.15% 1	32.57% 1
	Average ranks							
BR^{+}	1.68 1	4.41 5	2.50 1	6.36 6	4.68 4	2.14 1	5.59 5	4.32 5
$BR^{\pm}$	6.91 7	8.50 10	5.55 6	6.91 7	7.36 8.5	6.82 8	7.14 8	7.64 8
$SBR^{+}_{?}$	4.27 3	4.23 3	4.82 4	4.14 3	3.00 2	3.73 3	4.18 3	3.18 2
$SBR^{\pm}_{?}$	5.82 5	7.05 8	4.64 3	5.27 4	5.64 6	6.18 7	5.77 6	6.00 6
SBR^{+}_{d}	6.00 6	3.23 1	6.45 8	2.50 2	2.64 1	4.45 4	2.73 2	2.55 1
$SBR^{\pm}_{d}$	4.73 4	6.50 7	5.00 5	7.73 10	8.00 10	8.73 10	7.91 9	8.45 9
$SBR^{+}_{y/?}$	2.86 2	4.27 4	3.77 2	5.36 5	3.95 3	2.77 2	4.77 4	3.68 3
$SBR^{+}_{y/d}$	7.91 10	3.82 2	8.27 10	2.36 1	5.18 5	5.64 5	2.00 1	3.91 4
SBR^{+}_{Gibbs}	7.27 8	7.50 9	6.09 7	7.36 9	7.36 8.5	8.55 9	8.00 10	8.73 10
PCR	7.55 9	5.50 6	7.91 9	7.00 8	7.18 7	6.00 6	6.91 7	6.55 7

the stacked models allow labels which were initially missed to be found due to the label context.

The $SeCo^{\pm}$ variants seem to be quite competitive, especially when the post-optimization of Ripper is turned on. In addition, the performance appears to be robust particularly regarding the balancing between recall and precision. CAL500 is a clear exception. As we already could observe in Table 6, $SeCo^{\pm}$ only learns rules for a few of the labels. This could again stay in relation with inducing too general negative head rules, i.e., in this case, the default rule. On the other hand, also BR^{+} has problems on this dataset. Surprisingly, the second best performing approach regarding micro F1 is $SeCo^{+}$ (followed by $SBR^{\pm}_{?}$ and $SeCo^{+}_{opt}$) due to

its high recall although it has the lowest overall rank on micro F1 of the main approaches.

The high recall is inherent for the multi-label iterative separate-and-conquer approach when learning positive head rules. The reason is that each additional positive head rule in the decision lists can only increase the probability of predicting an additional label as true. Remind that processing the list will only stop when all labels were predicted or a rule marked as stopping rule applies. Increasing the τ sensibility for the stopping rule criteria, as well as turning off the re-insertion of full covered examples, may alleviate the problem. However, we did not try to specifically tune the parameters for every variant in our evaluation since we were more concerned with the produced models and the overall comparability of the approaches. We believe that tuning could certainly further improve the scores especially for the SeCo approaches.

Nevertheless, an informal comparison with the results published by [7] reveals that our approaches are likely to be highly competitive to other state-of-the-art and even more sophisticated multi-label approaches. We leave a thorough comparison of multi-label rule learning approaches to existing approaches for further work.

6.4.1 Additional Comparisons

The separated blocks in the result tables allow additional interesting comparisons, such as the comparison of SBR to its variant using only the label predictions as inputs. The objective was to determine the regret of having to rely only on label features, i.e., only on unconditional dependencies between the labels. From the results, we can observe that, in terms of micro F1, SBR^+ beats its counterpart SBR^+_y but only by a small gap. The margin is more pronounced for the non-absenting variants on F1. W.r.t. the other measures, $\text{SBR}^+_{y/d}$ has generally a high micro recall and $\text{SBR}^+_{y/?}$ usually wins at micro precision and the highly related Hamming accuracy. Both approaches beat their counterparts, respectively.

It is made clear by the experiments, that Gibbs sampling does not work well for rule models. This is most likely due to the higher sensibility of common rules to noise in discrete features compared to statistical approaches. Hence, the approach of Guo and Gu [21] may not work well for other symbolic approach, too.

Table 10 depicts results for the four larger datasets, namely MEDIAMILL, COREL16K, BIBTEX, and COREL5K. We can observe in general a higher tendency towards extreme results, especially regarding the trade-off between recall and precision, and also in general higher discrepancies between micro- and macro-averaged measures, such as for COREL5K. We suspect that the comparably high difficulty of the classification tasks together with the relatively low number of relevant labels to be matched compared to the high total amount of labels makes the outcome very sensitive towards peculiarities and noise of the datasets, leading to the high variance observed in the results.

However, compared to the previous results on the smaller datasets (cf. Tables 8 and 9), the general trends remain. BR^+ showed a small improvement for the seven small datasets in Hamming accuracy but was worse in macro F1 while remaining constant for all other measures. $\text{BR}^\pm$ was better for the small datasets in micro and macro precision and in micro F1, but worse in micro recall. While $\text{SBR}^\pm_?$ and SBR^+_d did not show noteworthy changes, $\text{SBR}^\pm_d$ interestingly changed for all used

measures. As it deteriorated only for Hamming and subset accuracy as well as for micro precision but improved for all other measures, it seems that this variant works better on smaller datasets. The reason presumably is that being able to also predict negative labels is only advantageous when the total number of labels is low. The more labels are contained in a dataset, the more these will be sparse, and therefore focusing only on the positive labels pays off much more. Consistent with this argument, we observed that for micro and macro F1, with only a few exceptions, it is more beneficial to predict only positive labels.

PCR, the only algorithm not relying on single-label rules and Ripper as base learner, can only reveal its advantages in few cases such as for YEAST on subset accuracy. These observations correspond to our expectation that single-label head rules allow a more compact representation of multi-label datasets (cf. Section 2.3). However, PCR improves compared to the other approaches for larger datasets on the macro-averaged measures. A possible reason might be that overfitting has been prevented implicitly due to the restriction of learning at most 1000 rules. However, a more exhaustive analysis would be necessary in order to draw definite conclusions due to the different problem domain for which PCR was designed and the different learning approach and heuristics used.

6.4.2 Statistical Tests

We performed statistical tests for all the different multi-label measures and algorithms shown in Tables 8 and 9. Following Demsar [13], first a Friedman test was conducted and secondly a post-hoc Nemenyi test. The Friedman test was passed for all measures ($p = 0.01$). The critical distance between average ranks, for which we can find a significant difference for $p = 0.01$, is 8.54 according to the Nemenyi test (14 algorithms on 7 datasets). The following significantly differing algorithm pairs were found: for Hamming accuracy only $\text{SeCo}^{\pm}_{opt}$ was significantly better than SeCo^{+}, while the other algorithms did not differ. For micro-averaged precision $\text{SeCo}^{\pm}_{opt}$ turned out to be better than $\text{SBR}^{+}_{y/d}$, SeCo^{+}_{opt}, and SeCo^{+}. However, for the macro-averaged variant of precision only BR^{+} outperformed SBR^{+}_{Gibbs}. The latter was significantly inferior for micro- and macro-averaged recall and F1-measure. For micro-averaged recall SeCo^{+} also significantly outperformed PCR. Both for micro- and macro-averaged F-Measure $\text{SBR}^{+}_{?}$ and SBR^{+}_{d} were better than SBR^{+}_{Gibbs}.

We also briefly show the statistical significance tests of the approaches except SeCo on all used datasets including the four larger ones. Again, the Friedman test passed and the critical distance was found to be 4.71 ($p = 0.01$). Regarding Hamming acccuracy, BR^{+} was significantly better than $\text{BR}^{\pm}$, SBR^{+}_{Gibbs}, PCR, and $\text{SBR}^{+}_{y/d}$ and $\text{SBR}^{+}_{y/?}$ outperformed $\text{SBR}^{+}_{y/d}$. For subset accuracy we observed that SBR^{+}_{d} is better than $\text{BR}^{\pm}$. BR^{+} outperformed PCR and $\text{SBR}^{+}_{y/d}$ for micro precision. For micro recall $\text{SBR}^{+}_{y/d}$ and SBR^{+}_{d} were superior to SBR^{+}_{Gibbs} and $\text{SBR}^{\pm}_{d}$. Micro F1 showed that $\text{SBR}^{+}_{?}$ significantly outperformed $\text{SBR}^{\pm}_{d}$ while SBR^{+}_{d} in addition also was better than $\text{BR}^{\pm}$ and SBR^{+}_{Gibbs}.

7 Conclusions

In this work, we introduced two approaches for making label dependencies explicit with the means of rules. Especially the proposed stacking approach is very effective at inducing rules with labels as conditions in the bodies of the rules. In our analyses on eleven multi-label datasets, the resulting models turned out to be indeed very useful in order to discover interesting aspects a normal, single-label rule learner is unable to uncover.

For instance, we found that the GENBASE dataset exhibits only very weak label dependencies, which can hardly be exploited in order to improve the predictive performance, despite the fact that this dataset is frequently used for evaluating multi-label algorithms. In contrast to other approaches, the proposed method naturally allows for discovering and expressing local as well as global label dependencies.

The multi-label iterative separate-and-conquer approach straightforwardly extends the well-known separate-and-conquer technique from a single target label to multiple labels. In contrast to applying separate-and-conquer separately for each of the labels, as for traditional concept learning, this technique not only allows to take dependencies into account, but also produces much more compact representations in form of multi-label decision lists, a generalization of common single-label multiclass decision lists.

However, the extensive evaluation on eleven benchmark datasets revealed the limitations of the integrated approach in discovering and exploiting label-dependent rules. This is mainly because multi-label iterative separate-and-conquer has to produce itself the label features it can later use, whereas, e.g., the stacking approach has all the needed label information available from the beginning. The experimental evaluation also showed that the stacking approach outperforms all remaining approaches w.r.t. to F1-measure. We hence believe that using stacking is particularly useful for exploiting global and local dependencies when using rule learners.

This work was limited to discovering label-dependent single-label head rules. Both approaches can learn this kind of multi-label rules. In addition, they can control to a certain degree the type of rule head and rule body. However, one of our future goals has to be to extent our approaches in order to also induce label-dependent multi-label head rules since only this kind of rules allows to express all types of possible dependencies. Our proposed iterative separate-and-conquer approach can be naturally extended in order to learn multiple labels in the heads. However, further research is needed first on possible improvements in the iterative algorithm as well as on the right selection of the heuristics for the local refinements. This is especially important considering that a key issue of multi-label classification are the different objective measures.

References

1. Aho, T., Ženko, B., Džeroski, S., Elomaa, T.: Multi-target regression with rule ensembles. Journal of Machine Learning Research 13(1), 2367–2407 (2012)
2. Aho, T., Zenko, B., Dzeroski, S.: Rule ensembles for multi-target regression. In: Proceedings of the Ninth IEEE International Conference on Data Mining (ICDM-09). pp. 21–30. IEEE (2009)

3. Alessandro, A., Corani, G., Mauá, D., Gabaglio, S.: An ensemble of bayesian networks for multilabel classification. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. pp. 1220–1225. AAAI Press (2013)
4. Allamanis, M., Tzima, F., Mitkas, P.: Effective Rule-Based Multi-label Classification with Learning Classifier Systems. In: Adaptive and Natural Computing Algorithms, 11th International Conference, ICANNGA 2013. pp. 466–476 (2013)
5. Ávila, J., Galindo, E., Ventura, S.: Evolving Multi-label Classification Rules with Gene Expression Programming: A Preliminary Study. In: Hybrid Artificial Intelligence Systems. vol. 6077, pp. 9–16. Springer (2010)
6. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press, 1st edn. (2010)
7. Chekina, L., Gutfreund, D., Kontorovich, A., Rokach, L., Shapira, B.: Exploiting label dependencies for improved sample complexity. Machine Learning 91(1), 1–42 (2013)
8. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning 76(2-3), 211–225 (2009)
9. Cohen, W.W.: Fast Effective Rule Induction. In: Proceedings of the 12th International Conference on Machine Learning (ICML-95). pp. 115–123 (1995)
10. D. Malerba, G.S., Esposito, F.: A multistrategy approach to learning multiple dependent concepts. In: Machine Learning and Statistics: The Interface, chap. 4, pp. 87–106 (1997)
11. 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-10). pp. 279–286. Omnipress (Jun 2010)
12. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning 88(1-2), 5–45 (2012)
13. Demsar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)
14. Fralick, S.C.: Learning to recognize patterns without a teacher. IEEE Transactions on Information Theory 13(1), 57–64 (1967)
15. Fürnkranz, J.: Separate-and-Conquer Rule Learning. Artificial Intelligence Review 13(1), 3–54 (February 1999)
16. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning 73(2), 133–153 (Jun 2008)
17. Fürnkranz, J., Widmer, G.: Incremental Reduced Error Pruning. In: Cohen, W., Hirsh, H. (eds.) Proceedings of the 11th International Conference on Machine Learning (ICML-94). pp. 70–77. Morgan Kaufmann, New Brunswick, NJ (1994)
18. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management. pp. 195–200. ACM (2005)
19. Gibaja, E., Ventura, S.: Multi-label learning: a review of the state of the art and ongoing research. Wiley Interdisciplinary Reviews: Data Mining and

Knowledge Discovery 4(6), 411–444 (2014)
20. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Advances in Knowledge Discovery and Data Mining (PAKDD 2004). pp. 22–30 (2004)
21. Guo, Y., Gu, S.: Multi-label classification using conditional dependency networks. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Two. pp. 1300–1305. IJCAI'11, AAAI Press (2011)
22. Huang, S.J., Yu, Y., Zhou, Z.H.: Multi-label hypothesis reuse. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 525–533. KDD '12, ACM, New York, NY, USA (2012)
23. Janssen, F., Fürnkranz, J.: On the quest for optimal rule learning heuristics. Machine Learning 78(3), 343–379 (Mar 2010)
24. Janssen, F., Zopf, M.: The SeCo-Framework for Rule Learning. In: Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2012 (2012)
25. Kumar, A., Vembu, S., Menon, A., Elkan, C.: Learning and inference in probabilistic classifier chains with beam search. In: Machine Learning and Knowledge Discovery in Databases, Proceedings of the ECML PKDD 2012, vol. 7523, pp. 665–680. Springer Berlin Heidelberg (2012)
26. Li, B., Li, H., Wu, M., Li, P.: Multi-label Classification based on Association Rules with Application to Scene Classification. In: Proceedings of the 2008 The 9th International Conference for Young Computer Scientists. pp. 36–41. IEEE Computer Society (2008)
27. Li, N., Zhou, Z.H.: Selective ensemble of classifier chains. In: Zhou, Z.H., Roli, F., Kittler, J. (eds.) Multiple Classifier Systems, vol. 7872, pp. 146–156. Springer Berlin Heidelberg (2013)
28. Li, Y.K., Zhang, M.L.: Enhancing binary relevance for multi-label learning with controlled label correlations exploitation. In: Pham, D.N., Park, S.B. (eds.) PRICAI 2014: Trends in Artificial Intelligence, Lecture Notes in Computer Science, vol. 8862, pp. 91–103. Springer International Publishing (2014)
29. Liu, B., Hsu, W., Yiming, M.: Integrating classification and association rule mining. In: Proceedings of the fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). pp. 80–86 (1998)
30. Loza Mencía, E., Janssen, F.: Stacking label features for learning multilabel rules. In: Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014, Proceedings, Lecture Notes in Computer Science, vol. 8777, pp. 192–203. Springer (2014)
31. Madjarov, G., Gjorgjevikj, D., Delev, T.: Efficient two stage voting architecture for pairwise multi-label classification. In: Li, J. (ed.) AI 2010: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 6464, pp. 164–173. Springer Berlin / Heidelberg (2011)
32. Madjarov, G., Gjorgjevikj, D., Džeroski, S.: Two stage architecture for multi-label learning. Pattern Recognition 45(3), 1019 – 1034 (Mar 2012)
33. Madjarov, G., Kocev, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9), 3084 – 3104 (2012)
34. McCallum, A.K.: Multi-label text classification with a mixture model trained by EM. In: AAAI 99 Workshop on Text Learning (1999)

35. Montañés, E., Senge, R., Barranquero, J., Quevedo, J.R., del Coz, J.J., Hüllermeier, E.: Dependent binary relevance models for multi-label classification. Pattern Recognition 47(3), 1494 – 1508 (2014)
36. Papagiannopoulou, C., Tsoumakas, G., Tsamardinos, I.: Discovering and exploiting deterministic label relationships in multi-label learning. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 915–924. KDD '15, ACM, New York, NY, USA (2015)
37. Park, S.H., Fürnkranz, J.: Multi-label classification with label constraints. In: Hüllermeier, E., Fürnkranz, J. (eds.) Proceedings of the ECML PKDD 2008 Workshop on Preference Learning (PL-08, Antwerp, Belgium). pp. 157–171 (2008)
38. Pestian, J., Brew, C., Matykiewicz, P., Hovermale, D., Johnson, N., Cohen, K.B., Duch, W.: A shared task involving multi-label classification of clinical free text. In: Proceedings of the Workshop on BioNLP 2007 at ACL 2007 (2007)
39. Read, J., Martino, L., Luengo, D.: Efficient monte carlo methods for multi-dimensional learning with classifier chains. Pattern Recognition 47(3), 1535–1546 (2014)
40. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning 85(3), 333–359 (2011)
41. Scudder, III, H.: Probability of error of some adaptive pattern-recognition machines. IEEE Trans. Inf. Theor. 11(3), 363–371 (Sep 2006)
42. Senge, R., del Coz, J.J., Hüllermeier, E.: Rectifying classifier chains for multi-label classification. In: Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2013, pp. 162–169 (2013)
43. Stecher, J., Janssen, F., Fürnkranz, J.: Separating Rule Refinement and Rule Selection Heuristics in Inductive Rule Learning. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-14), Part 3. Lecture Notes in Computer Science, vol. 8726, pp. 114–129. Springer, Nancy, France (Sep 2014)
44. Sucar, L.E., Bielza, C., Morales, E.F., Hernandez-Leal, P., Zaragoza, J.H., Larraaga, P.: Multi-label classification with bayesian network-based chain classifiers. Pattern Recognition Letters 41, 14 – 22 (2014)
45. Thabtah, F., Cowling, P., Peng, Y.: MMAC: A New Multi-Class, Multi-Label Associative Classification Approach. In: Proceedings of the 4th IEEE ICDM. pp. 217–224 (2004)
46. Tsoumakas, G., Katakis, I.: Multi Label Classification: An Overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
47. Tsoumakas, G., Dimou, A., Spyromitros Xioufis, E., Mezaris, V., Kompatsiaris, I., Vlahavas, I.P.: Correlation based pruning of stacked binary relevance models for multi-label learning. In: Proc 1st Int Workshop on Learning from Multi-Label Data (MLD'09). pp. 101–116 (2009)
48. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining Multi-label Data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685 (2010)
49. Veloso, A., Meira, W., Gonçalves, M.A., Zaki, M.: Multi-label lazy associative classification. In: Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. pp. 605–612 (2007)

50. Ženko, B., Džeroski, S.: Learning classification rules for multiple target attributes. In: Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2008). vol. 5012, pp. 454–465. Springer (2008)
51. Witten, I.H., Frank, E.: Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, 2nd edn. (2005)
52. Zhang, M.L., Zhang, K.: Multi-label learning by exploiting label dependency. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 999–1008. ACM (2010)
53. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. Knowledge and Data Engineering, IEEE Transactions on 26(8), 1819–1837 (Aug 2014)
54. Zhu, S., Ji, X., Xu, W., Gong, Y.: Multi-labelled classification using maximum entropy method. In: SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 274–281. ACM (2005)