Multi-label LeGo — Enhancing Multi-label Classifiers with Local Patterns

Wouter Duivesteijn[1], Eneldo Loza Mencía[2], Johannes Fürnkranz[2], and Arno Knobbe[1]

[1] LIACS, Leiden University, the Netherlands, {wouterd,knobbe}@liacs.nl
[2] Knowledge Engineering Group, TU Darmstadt, Germany, {eneldo,juffi}@ke.tu-darmstadt.de

Abstract. The straightforward approach to multi-label classification is based on decomposition, which essentially treats all labels independently and ignores interactions between labels. We propose to enhance multi-label classifiers with features constructed from local patterns representing explicitly such interdependencies. An Exceptional Model Mining instance is employed to find local patterns representing parts of the data where the conditional dependence relations between the labels are exceptional. We construct binary features from these patterns that can be interpreted as partial solutions to local complexities in the data. These features are then used as input for multi-label classifiers. We experimentally show that using such constructed features can improve the classification performance of decompositive multi-label learning techniques.

Keywords: Exceptional Model Mining; Multi-Label Classification; LeGo

1 Introduction

Contrary to ordinary classification, in multi-label classification (MLC) one can assign more than one class label to each example [1]. For instance, when we have the earth's continents as classes, a news article about the French and American interference in Libya could be labeled with the *Africa*, *Europe*, and *North America* classes. Originally, the main motivation for the multi-label approach came from the fields of medical diagnosis and text categorization, but nowadays multi-label methods are required by applications as diverse as music categorization, semantic scene classification, and protein function classification.

Many approaches to MLC take a decompositive approach, i.e., they decompose the MLC problem into a series of ordinary classification problems. The formulation of these problems often ignores interdependencies between labels, implying that the predictive performance may improve if label dependencies are taken into account. When, for instance, one considers a dataset where each label details the presence or absence of one kind of species in a certain region, the food chains between the species cause a plethora of strong correlations between labels. But interplay between species is more subtle than just correlations between pairs of species. It has, for instance, been shown [2] that a food chain between two

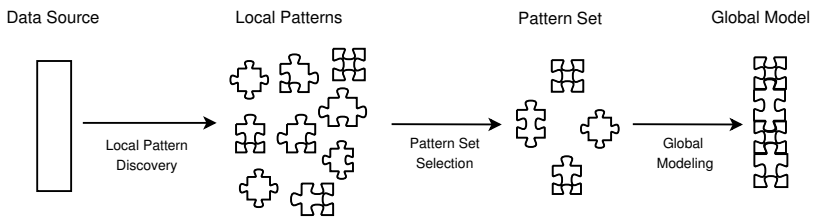

Fig. 1. The LeGo framework

species (the sponge *Haliclona* and the nudibranch *Anisodoris*) may be displaced depending on whether a third species is present (the starfish *Pisaster ochraceus*), which is not directly related to the species in the food chain. Apparently, there is some conditional dependence relation between these three species. The ability to consider such interplay is an essential element of good multi-label classifiers.

In this paper we propose incorporating locally exceptional interactions between labels in MLC, as an instance of the LeGo framework [3]. In this framework, the KDD process is split up in several phases: first local models are found each representing only part of the data, then a subset of these models is selected, and finally this subset is employed in constructing a global model. The crux is that straight-forward classification methods can be used for building a global classifier, if the locally exceptional interactions between labels are represented by features constructed from patterns found in the local modeling phase.

We propose to find patterns representing these locally exceptional interactions through an instance of Exceptional Model Mining [4]; a framework that can be seen as an extension of traditional Subgroup Discovery. The instance we consider [5] models the conditional dependencies between the labels by a Bayesian network, and strives to find patterns for which the learned network has a substantially different structure than the network learned on the whole dataset. These patterns can each be represented by a binary feature of the data, and the main contribution of this paper is a demonstration that the integration of these features into the classification process improves classifier performance.

2 Preliminaries

In this section, we recall the cornerstones of our work: the LeGo framework for learning global models from local patterns (Section 2.1) and multi-label classification (Section 2.2). We conclude with the problem formulation (Section 2.3).

2.1 The LeGo framework

As mentioned, the work in this paper relies heavily on the LeGo framework [3]. This framework assumes that the induction process is not executed by running a single learning algorithm, but rather consists of a number of consecutive phases,

as illustrated in Figure 1. In the first phase a local pattern discovery algorithm is employed in order to obtain a number of informative patterns, which can serve as relevant features to be used in the subsequent phases. These patterns can be considered partial solutions to local complexities in the data. In the second and third phase, the patterns are filtered to reduce redundancy, and the selected patterns are combined in a final global model, which is the outcome of the process.

The main reason to invest the additional computational cost of a LeGo approach over a single-step algorithm, is the expected increase in accuracy of the final model, caused by the higher level of exploration involved in the initial local pattern discovery phase. Typically, global modeling techniques employ some form of greedy search, and in complex tasks, subtle interactions between attributes may be overlooked as a result of this. In most pattern mining methods however, extensive consideration of combinations of attributes is quite common. When employing such exploratory algorithms as a form of preprocessing, one can think of the result (the patterns) as partial solutions to local complexities in the data. The local patterns, which can be interpreted as new virtual features, still need to be combined into a global model, but potentially hard aspects of the original representation will have been accounted for. As a result, straightforward methods such as Support Vector Machines with linear kernels can be used in the global modeling phase.

The LeGo approach has shown its value in a range of settings [3], particularly regular binary classification [6, 7], but we have specific reasons for choosing this approach in the context of multi-label classification (MLC). It is often mentioned that in MLC, one needs to take into consideration potential interactions between the labels, and that simultaneous classification of the labels may benefit from knowledge about such interactions [8, 9].

In a previous publication [5], we have outlined an algorithm finding local interactions amongst multiple targets (labels) by means of an Exceptional Model Mining (EMM) instance. The EMM framework [4] suggests a discovery approach involving multiple targets, using local modeling over the targets in order to find subsets of the dataset where unusual (joint) target distributions can be observed. In [5], we presented one instance of EMM that deals with discrete targets, and employs Bayesian Networks in order to find patterns corresponding to unusual dependencies between targets. This Bayesian EMM instance quenches the thirst in MLC for representations of locally unusual combinations of labels.

2.2 Multi-label classification

Throughout this paper we assume a dataset Ω. This is a bag of N elements (*data points*) of the form $x = \{a_1, \ldots, a_k, \ell_1, \ldots, \ell_m\}$, where k and m are positive integers. We call $a_1, \ldots, a_k$ the *attributes* of x, and $\ell_1, \ldots, \ell_m \in \mathcal{L}$ the *labels* of x. Each label ℓ_i is assumed to be discrete, and the vectors of attributes are taken from an unspecified domain $\mathcal{A}$. Together we call the attributes and labels of x the *features* of x. When necessary, we distinguish the ith data point from other data points by adding a superscript i to the relevant symbols.

Attributes	Labels $\in \{0,1\}^m$
$a_1^1, \ldots, a_k^1$	$\ell_1^1, \ldots, \ell_m^1$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n$	$\ell_1^n, \ldots, \ell_m^n$

(a) input training set

Attributes	Class $\in 2^{\mathcal{L}}$
$a_1^1, \ldots, a_k^1$	$\mathcal{Y}^1$
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	$\mathcal{Y}^n$

(b) *label powerset* (LP) decomposition

Attributes	Class$_1 \in \{0,1\}$
$a_1^1, \ldots, a_k^1$	ℓ_1^1
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	ℓ_1^n

$\cdots$

Attributes	Class$_m \in \{0,1\}$
$a_1^1, \ldots, a_k^1$	ℓ_m^1
$\vdots \quad \ddots \quad \vdots$	$\vdots$
$a_1^n, \ldots, a_k^n$	ℓ_m^n

(c) *binary relevance* (BR) decomposition

Fig. 2. Decomposition of multi-label training sets into binary (BR) or multiclass problems (LP). $\mathcal{Y}^i = \{y_1^i, \ldots, y_{|\mathcal{Y}^i|}^i |\ y_j^i \in \mathcal{L}\}$ denotes the assigned labels $\{\ell_j |\ \ell_j^i = 1\}$ to example x^i. In LP the (single) target value of an instance x^i is from the set $\{\mathcal{Y}^i |\ i = 1 \ldots m\} \subseteq 2^{\mathcal{L}}$ of the different label subsets seen in the training data.

The task of multi-label classification (MLC) is, given a training set $\mathcal{E} \subset \Omega$, to learn a function $f(a_1, \ldots, a_k) \rightarrow (\ell_1, \ldots, \ell_m)$ which predicts the labels for a given example. Many multi-label learning techniques reduce this problem to ordinary classification. The widely used *binary relevance* (BR) [1] approach tackles a multi-label problem by learning a separate classifier $f_i(a_1, \ldots, a_k) \rightarrow \ell_i$ for each label ℓ_i, as illustrated in Figure 2c. At query time, each binary classifier predicts whether its class is relevant for the query example or not, resulting in a set of relevant labels. Obviously, BR ignores possible interdependencies between classes since it learns the relevance of each class independently. One way of addressing this problem is by using *classifier chains* (CC) [9], which are able to model label dependencies since they stack the outputs of the models: the prediction of the model for label ℓ_i depends on the predictions for labels $\ell_1, \ldots, \ell_{i-1}$.

An alternative approach is *calibrated label ranking* (CLR) [10], where the key idea is to learn one classifier for each binary comparison of labels. CLR learns binary classifiers $f_{ij}(a_1, \ldots, a_k) \rightarrow (\ell_i \succ \ell_j)$, which predict for each label pair (ℓ_i, ℓ_j) whether ℓ_i is more likely to be relevant than ℓ_j. Thus, CLR (implicitly) takes correlations between pairs of labels into account. In addition, the decomposition into pairs of classes has the advantage of simpler sub-problems and hence commonly more accurately performing models. Finally, a simple way to take label dependencies into account is the *label powerset* (LP) approach [1], treating each combination of labels occuring in the training data as a separate label of a classification problem (Figure 2b).

We will use each of these techniques for decomposing a multi-label problem into an ordinary classification problem in the third LeGo phase (Section 4).

2.3 Problem statement

The main question this paper addresses is whether a LeGo approach can improve multi-label classification, compared to existing methods that do not employ a preliminary local pattern mining phase. Thus, our approach encompasses:

1. find a set P of patterns representing local anomalies in conditional dependence relations between labels, using the method introduced in [5];
2. filter out a meaningful subset $S \subseteq P$;
3. use the patterns in S as constructed features to enhance multi-label classification methods.

In this paper we will use sophisticated methods in phase 1 and 3. In phase 2, we simply draw S as a random sample of P. The following two sections will explore what we do in phases 1 and 3.

3 Local Pattern Discovery phase

To find the local patterns with which we will enhance the MLC feature set, we employ an instance of Exceptional Model Mining (EMM). This instance is tailored to find subgroups in the data where the conditional dependence relations between a set of target features (our labels) is significantly different from those relations on the whole dataset. Before we recall the EMM instance in more detail, we will outline the general EMM framework.

3.1 Exceptional Model Mining

Exceptional Model Mining is a framework that can be considered an extension of the traditional Subgroup Discovery (SD) framework, a supervised learning task which strives to find *patterns* (defined on the input variables) that satisfy a number of user-defined *constraints*. A pattern is a function $p : \mathcal{A} \to \{0, 1\}$, which is said to *cover* a data point x^i if and only if $p\left(a_1^i, \ldots, a_k^i\right) = 1$. We refer to the set of data points covered by a pattern p as the *subgroup* corresponding to p. The *size* of a subgroup is the number of data points the corresponding pattern covers. The user-defined constraints typically include lower bounds on the subgroup size and on the quality of the pattern, which is usually defined on the output variables. A run of an SD algorithm results in a quality-ranked list of patterns satisfying the user-defined constraints.

In traditional SD, we have only a single target variable. The quality of a subgroup is typically gauged by weighing its target distribution deviation and its size. EMM extends SD by allowing for more complex target concepts defined on multiple target variables. It partitions the features into two sets: the attributes and the labels. On the labels a model class is defined, and an exceptionality

measure φ for that model class is selected. Such a measure assigns a quality value $\varphi(p)$ to a candidate pattern p. EMM algorithms traverse a search lattice of candidate patterns, constructed on the attributes, in order to find patterns that have exceptional values of φ on the labels.

3.2 Exceptional Model Mining meets Bayesian networks

As discussed in Section 2, we assume a partition of the $k + m$ features in our dataset into k attributes, which can be from any domain, and m labels, which are assumed to be discrete. The EMM instance we employ [5] proposes to use Bayesian networks (BNs) over those m labels as model class. These networks are directed acyclic graphs (DAGs) that model the conditional dependence relations between their nodes. A pattern has a model that is exceptional in this setting, when the conditional dependence relations between the m labels are significantly different on the data covered by the pattern than on the whole dataset. Hence the exceptionality measure needs to measure this difference. We will employ the Weighed Entropy and Edit Distance measure (denoted φ_{weed}), as introduced in [5]. This measure indicates the extent to which the BNs differ in structure. Because of the peculiarities of BNs, we cannot simply use traditional edit distance between graphs [11] here. Instead, a variant of edit distance for BNs was introduced, that basically counts the number of violations of the famous theorem by Verma and Pearl on the conditions for equivalence of DAG models [12]:

Theorem 1 (Equivalent DAGs). *Two DAGs are equivalent if and only if they have the same skeleton and the same v-structures.*

Since these two conditions determine whether two DAGs are equivalent, it makes sense to consider the number of differences in skeletons and v-structures as a measure of how different two DAGs are. For more details on φ_{weed}, see [5].

After running the EMM algorithm, we obtain a set P of patterns each representing a local exceptionality in the conditional dependence relations between the m labels, hence completing the Local Pattern Discovery phase.

4 Global Modeling phase

As stated in Section 2.3, we do nothing sophisticated in the Pattern Set Selection phase. Instead, we filter out a pattern subset $S \subseteq P$ by taking a random sample from P. In the current section, we use this subset S to learn a global model.

For the learning of the global multi-label classification models in the Global Modeling phase, we experiment with standard approaches including binary relevance (BR) and label powerset (LP) decompositions [1], as well as effective recent state-of-the-art learners such as calibrated label ranking (CLR) [10], and classifier chains (CC) [9]. The chosen algorithms cover a wide range of approaches and techniques used for learning multi-label problems (see Section 2.2), and are all included in Mulan, a library for multi-label classification algorithms [1].

Attributes	Labels
$a_1^1, \ldots, a_k^1$	$\ell_1^1, \ldots, \ell_m^1$
$a_1^2, \ldots, a_k^2$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n$	$\ell_1^n, \ldots, \ell_m^n$

(a) input training set C_O

Attributes	Labels
$p_1\left(a_1^1, \ldots, a_k^1\right), \ldots, p_{\lvert S\rvert}\left(a_1^1, \ldots, a_k^1\right)$	$\ell_1^1, \ldots, \ell_m^1$
$p_1\left(a_1^2, \ldots, a_k^2\right), \ldots, p_{\lvert S\rvert}\left(a_1^2, \ldots, a_k^2\right)$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$p_1\left(a_1^n, \ldots, a_k^n\right), \ldots, p_{\lvert S\rvert}\left(a_1^n, \ldots, a_k^n\right)$	$\ell_1^n, \ldots, \ell_m^n$

(b) transformation into pattern space C_S

Attributes	Labels
$a_1^1, \ldots, a_k^1, p_1\left(a_1^1, \ldots, a_k^1\right), \ldots, p_{\lvert S\rvert}\left(a_1^1, \ldots, a_k^1\right)$	$\ell_1^1, \ldots, \ell_m^1$
$a_1^2, \ldots, a_k^2, p_1\left(a_1^2, \ldots, a_k^2\right), \ldots, p_{\lvert S\rvert}\left(a_1^2, \ldots, a_k^2\right)$	$\ell_1^2, \ldots, \ell_m^2$
$\vdots \quad \ddots \quad \vdots \quad \vdots \quad \ddots \quad \vdots$	$\vdots \quad \ddots \quad \vdots$
$a_1^n, \ldots, a_k^n, p_1\left(a_1^n, \ldots, a_k^n\right), \ldots, p_{\lvert S\rvert}\left(a_1^n, \ldots, a_k^n\right)$	$\ell_1^n, \ldots, \ell_m^n$

(c) combined attributes in the LeGo classifier C_L

Fig. 3. A multi-label classification problem (a), its representation in pattern space (b) given the set of patterns $p_1, \ldots, p_{|S|}$, and the LeGo combination (c)

For each classifier configuration, we learn three classifiers based on different feature sets. The first classifier uses the k features that make up the original dataset, and is denoted C_O (Figure 3a). The second classifier, denoted C_S, uses features constructed from our pattern set S. Each of these patterns maps each record in the original dataset to either zero or one. Hence they can be trivially transformed into binary features, that together make up the feature space for classifier C_S (Figure 3b). The third classifier employs both the k original and $|S|$ constructed features, in the spirit of LeGo, and is hence denoted C_L.

5 Experimental setup

To experimentally validate the outlined LeGo method, we will compare the performance of the three classifiers based on different feature sets C_O, C_S, and C_L. We will also investigate the relative performance of the different feature selection methods, and the relative performance of classification approaches.

For the experiments we selected three multi-labeled datasets from different domains. Statistics on these datasets can be found in Table 1. The column *Cardinality* displays the average number of relevant labels for a data point.

We combine the multilabel decomposition methods mentioned in Section 4 with several base learners: J48 with default settings [13], standard LibSVM [14], and LibSVM with a grid search on the parameters. In this last approach, multiple values for the SVM kernel parameters are tried, and the one with the best 3-fold cross-validation accuracy is selected for learning on the training set (as

Table 1. Datasets used in the experiments, shown with the number of examples (N), attributes (k), and labels (m), as well as the average number of labels per example

Dataset	Domain	N	k	m	Cardinality
Emotions	Music	593	72	6	1.87
Scene	Vision	2407	294	6	1.07
Yeast	Biology	2417	103	14	4.24

suggested by [14]). Both SVM methods are run once with the Gaussian Radial Basis Function as kernel, and once with a linear kernel using the efficient LibLinear implementation [15]. We will refer to LibSVM with the parameter grid search as MetaLibSVM, and denote the used kernel by a superscript *rbf* or *lin*.

5.1 Experimental procedure

All statistics on the classification processes are estimated via a 10-fold cross-validation. To enable a fair comparison of the LeGo classifier with the other classifiers, we let the entire learning process consider only the training set for each fold. This means that we have to run the Local Pattern Discovery and Pattern Subset Discovery phase separately for each fold.

For every fold on every dataset, we determine the best 10,000 patterns, measuring the exceptionality with φ_{weed} as described in Section 3.2. The search space in EMM cannot be explored exhaustively when there are numerical attributes and a nontrivial quality measure, and both are the case here. Hence we resort to a beam search strategy, configured with a beam width of $w = 10$ and a maximum search level of 2 (for more details on beam search in EMM, see [5]). We specifically select a search of modest depth, in order to prevent producing an abundance of highly similar patterns. We further bound the search by setting the minimal coverage of a pattern at 10% of the dataset.

For each dataset for each fold, we train classifiers from the three training sets C_O, C_S, and C_L for each combination of a decomposition approach and base learner. We randomly select $|S| = k$ patterns (cf. Section 4), i.e. exactly as many pattern-based features for C_S and C_L as there are original features in C_O.

5.2 Evaluation measures

We evaluate the effectiveness of the three classifiers for each combination on the respective test sets for each fold with five measures: Micro-Averaged Precision and Recall, Subset Accuracy, Ranking Loss, and Average Precision (for details on computation cf. [10] and [1]). We find these five measures a well balanced selection from the vast set of multi-label measures, evaluating different aspects of multi-label predictions such as good ranking performance and correct bipartition.

From a confusion matrix aggregated over all labels and examples, *Precision* computes the percentage of predicted labels that are relevant, and *Recall* computes the percentage of relevant labels that are predicted. *Subset Accuracy*

Table 2. Average ranks r_i of the three classifiers $C_i, i \in \{O, S, L\}$, with critical difference CD, over all test configurations, and over all test configurations barring J48

	r_O	r_S	r_L	CD
Overall	1.863	2.340	1.797	0.191
Without J48	1.971	2.296	1.733	0.214

denotes the percentage of perfectly predicted label sets, basically forming a multi-label version of traditional accuracy. We also computed the following rank-based loss measures. *Ranking Loss* returns the number of pairs of labels which are not correctly ordered, normalized by the total number of pairs. *Average Precision* computes the precision at each relevant label in the ranking, and averages these percentages over all relevant labels. These two ranking measures are computed for each example and then averaged over all examples.

All values for all settings are averaged over the folds of the cross-validation. Thus we obtain 300 test cases (5 evaluation measures $\times$ 5 base learners $\times$ 4 decomposition approaches $\times$ 3 datasets). To draw conclusions from these raw results, we use the Friedman test with post-hoc Nemenyi test [16].

6 Experimental Results

Table 2 compares the three different representations C_O, C_S, and C_L over the grand total of 300 test cases in terms of average ranks.[3] We see that both C_O and C_L perform significantly ($\alpha = 5\%$)[4] better than C_S, i.e. the pattern-only classifier cannot compete with the original features or the combined classifier. However, when we consider only the LibSVM$^{\text{rbf}}$ base learner, we find that the pattern-only classifier outperforms the classifier trained on original features.

The difference in performance between C_O and C_L is not significant. Although the average rank for the LeGo-based classifier is somewhat higher, we cannot claim that adding local patterns leads to a significant improvement. However, when splitting out the results for the different base learners, we notice a striking difference in average ranks between J48 and the rest. When we restrict ourselves to the results obtained with J48, we find that $r_O = 1.433$, $r_S = 2.517$, and $r_L = 2.050$, with $CD = 0.428$. Here, the classifier built from original features significantly ($\alpha = 5\%$) outperforms the LeGo classifier.

[3] The results were consistent over all 5 measures with respect to the used feature sets so we did not further differentiate.

[4] Since we are comparing three classifiers, the Friedman statistic equals $\chi_F^2 = 52.687$. With significance level $\alpha = 5\%$, the critical value for the chi-squared distribution with 2 degrees of freedom equals 5.991, hence the null hypothesis of the Friedman test is comfortably rejected. For the post-hoc Nemenyi test, when comparing three classifiers the critical value is $q_{0.05} = 2.344$. Hence, the critical difference between average ranks becomes $CD = 0.191$, with significance level $\alpha = 5\%$.

One reason for the performance gap between J48 and the SVM approach lies in the way these approaches construct their decision boundary. The SVM approaches draw one hyperplane through the attribute space, whereas J48 constructs a decision tree, which corresponds to a decision boundary consisting of axis-parallel fragments. The patterns the EMM algorithm finds in the Local Pattern Discovery phase are constructed by several conditions on single attributes. Hence the domain of each pattern has a shape similar to a J48 decision boundary, unlike a (non-degenerate) SVM decision boundary. Hence, the expected performance gain when adding such local patterns to the attribute space is much higher for the SVM approaches than for the J48 approach.

Because the J48 approach results in such deviating ranks, we investigate the relative performance of the base learners. We compare their performance on the three classifiers C_O, C_S, and C_L, with decomposition methods BR, CC, CLR, and LP, on the datasets from Table 1, evaluated with the measures introduced in Section 5.1. The average ranks of the base learners over these 180 test cases can be found in Table 3; J48 performs significantly worse than all SVM methods.

We have determined that the performance difference between C_O and C_L is not significant. In order to see if we can make a weaker statement of significance between C_O and C_L, and having just established that this is the worst-performing base learner, we repeat our comparison of the classifiers C_O, C_S, and C_L on the four base learner approaches that perform best: the SVM variants. The average ranks of the three classifiers on these 240 test cases can be found in the last row of Table 2. On the SVM methods, the LeGo classifier is significantly ($\alpha = 5\%$) better than the classifier built from original features.

As stated in Section 2.2, to predict label ℓ_i the CC decomposition approach allows using the predictions made for labels $\ell_1, \ldots, \ell_{i-1}$. Hence we can view CC as a feature enriching approach, adding a feature set C. We find that adding C has an effect on performance similar to adding S, which is amplified when both are added, particularly for BR. Hence the patterns in S provide additional information on the label dependencies which is not covered by C.

7 Conclusions

We have proposed enhancing multi-label classification methods with local patterns in a LeGo setting. These patterns are found through an instance of Exceptional Model Mining, a generalization of Subgroup Discovery striving to find subsets of the data with aberrant conditional dependence relations between target features. Hence each pattern delivered represents a local anomaly in conditional dependence relations between targets. Each pattern corresponds to a binary feature which we add to the dataset, to improve classifier performance.

Experiments on three datasets show that for multi-label SVM classifiers the performance of the LeGo approach is significantly better than the traditional classification performance: investing extra time in running the EMM algorithm pays off when the resulting patterns are used as constructed features. The J48 classifier does not benefit from the local pattern addition, which can be at-

Table 3. Average ranks of the base learners, with critical difference CD

Approach	MetaLibSVM$^{\text{rbf}}$	MetaLibSVM$^{\text{lin}}$	LibSVM$^{\text{lin}}$	LibSVM$^{\text{rbf}}$	J48	CD
Rank	1.489	2.972	3.228	3.417	3.894	0.455

tributed to the similarity of the local decision boundaries produced by the EMM algorithm to those produced by the decision tree learner. Hence the expected performance gain when adding local patterns is lower for J48 than for approaches that learn different types of decision boundaries, such as SVM approaches.

The Friedman-Nemenyi analysis also shows that the constructed features generally cannot *replace* the original features without significant loss in classification performance. We find this reasonable, since these features are constructed from patterns found by a search process that is not at all concerned with the potential of the patterns for classification, but is focused on exceptionality. In fact, the pattern set may be highly redundant. Additionally it is likely that the less exceptional part of the data, which by definition is the majority of the dataset, is underrepresented by the constructed features.

To the best of our knowledge, this is a first shot at discovering multi-label patterns and testing their utility for classification in a LeGo setting. Therefore this work can be extended in various ways. It might be interesting to develop more efficient techniques without losing performance. One could also explore other quality measures, such as the plain edit distance measure from [5], or other search strategies. In particular, optimizing the beam-search in order to properly balance its levels of exploration and exploitation, could fruitfully produce a more diverse set of features [17] in the Local Pattern Discovery phase. Alternatively, pattern diversity could be addressed in the Pattern Subset Selection phase, ensuring diversity within the subset S rather than enforcing diversity over the whole pattern set P.

As future work, we would like to expand our evaluation of these methods. Recently, it has been suggested that for multi-label classification, it is better to use stratified sampling than random sampling when cross-validating [18]. Also, experimentation on more datasets seems prudent. In this paper, we have experimented on merely three datasets, selected for having a relatively low number of labels. As stated in Section 3.2, we have to fit a Bayesian network on the labels for each subgroup under consideration, which is a computationally expensive operation. The availability of more datasets with not too many labels (say, $m < 50$) would allow for more thorough empirical evaluation, especially since it would allow us to draw potentially significant conclusions from Friedman and Nemenyi tests per evaluation measure per base classifier per decomposition scheme. With three datasets this would be impossible, so we elected to aggregate all these test cases in one big test. The observed consistent results over all evaluation measures provide evidence that this aggregation is not completely wrong, but theoretically this violates the assumption of the tests that all test cases are inde-

pendent. Therefore, the empirically drawn conclusions in this paper should not be taken as irrefutable proof, but more as evidence contributing to our beliefs.

Acknowledgments This research is financially supported by the Netherlands Organisation for Scientific Research (NWO) under project number 612.065.822 (Exceptional Model Mining) and by the German Science Fundation (DFG). We also gratefully acknowledge support by the Frankfurt Center for Scientific Computing.

References

1. G. Tsoumakas, I. Katakis, I. P. Vlahavas, Mining Multi-label Data, Data Mining and Knowledge Discovery Handbook, Springer, pp. 667–685, 2010.
2. R. T. Paine, Food Web Complexity and Species Diversity, The American Naturalist 100 (910), pp. 65–75, 1966.
3. J. Fürnkranz, A. Knobbe (eds.), Special Issue: Global Modeling Using Local Patterns, Data Mining and Knowledge Discovery journal 20 (1), 2010.
4. D. Leman, A. Feelders, A. Knobbe, Exceptional Model Mining, Proc. ECML/PKDD (2), pp. 1–16, 2008.
5. W. Duivesteijn, A. Knobbe, A. Feelders, M. van Leeuwen, Subgroup Discovery meets Bayesian networks – an Exceptional Model Mining approach, Proc. ICDM, pp. 158–167, 2010.
6. A. Knobbe, J. Valkonet, Building Classifiers from Pattern Teams, From Local Patterns to Global Models: Proc. ECML PKDD 2009 Workshop, Slovenia, 2009.
7. J.-N. Sulzmann, J. Fürnkranz, A Comparison of Techniques for Selecting and Combining Class Association Rules, From Local Patterns to Global Models: Proc. ECML PKDD 2008 Workshop, Belgium, 2008.
8. W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Machine Learning 76 (2-3), pp. 211–225, 2009.
9. J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier Chains for Multi-label Classification, Proc. ECML PKDD, pp. 254–269, Springer, 2009.
10. J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, K. Brinker, Multilabel Classification via Calibrated Label Ranking, Machine Learning 73 (2), pp. 133–153, 2008.
11. L. G. Shapiro, R. M. Haralick, A Metric for Comparing Relational Descriptions, IEEE Trans. Pattern Anal. Mach. Intell. 7, pp. 90–94, 1985.
12. T. Verma, J. Pearl, Equivalence and Synthesis of Causal Models, Proc. UAI, pp. 255–270, 1990.
13. I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, San Francisco, 2005.
14. C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
15. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9, 1871–1874, 2008.
16. J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7, pp. 1–30, 2006.
17. M. van Leeuwen, A. J. Knobbe, Non-Redundant Subgroup Discovery in Large and Complex Data, Proc. ECML PKDD (3), pp. 459–474, 2011.
18. K. Sechidis, G. Tsoumakas, I. P. Vlahavas, On the Stratification of Multi-label Data, Proc. ECML PKDD (3), pp. 145–158, 2011.