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 [2, 3]. For instance, when we have the earth's continents as classes, a news article about the French and American intervention 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 [4], semantic scene classification [5], and protein function classification [6].

* This Technical Report, available at `http://www.ke.tu-darmstadt.de/publications/reports/tud-ke-2012-02.pdf`, is a longer version of the article with the same title [1] which appeared in J. Hollmen et al. (Eds.): *Advances in Intelligent Data Analysis XI 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings*, LNAI 7619, pp. 114-125, 2012, `http://www.springerlink.com/content/0635m1626m3j7866/` and which is also freely available as authors' version at `http://www.ke.tu-darmstadt.de/publications/papers/IDA-12.pdf`.

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 [7] that a food chain between two 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 [8,9]. 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 [10, 11]; a framework that can be seen as an extension of traditional Subgroup Discovery. The instance we consider [12] 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. The main contribution of this paper is a demonstration that the integration of these features into the classification process improves classifier performance. On the other hand, we expect the newly generated binary features to be expressive enough to replace the original features, while maintaining classifier performance and increasing efficiency.

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 [8,9]. This framework assumes that the induction process is not executed by running a

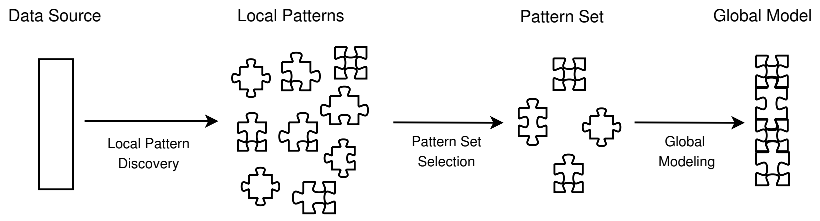

Fig. 1. The LeGo framework

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 [8], particularly regular binary classification [13,14], 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 [15–18].

In [12], an algorithm was outlined finding local interactions amongst multiple targets (labels) by means of an Exceptional Model Mining (EMM) instance. The EMM framework [11] 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 [12], we presented one instance of EMM that deals with discrete targets, and employs Bayesian

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

(a) input training set

Attributes	Class $\in \mathcal{L}$
$a_1^1, \ldots, a_k^1$	y_1^1
$\vdots \ \ddots \ \vdots$	$\vdots$
$a_1^1, \ldots, a_k^1$	$y_{\lvert\mathcal{Y}^1\rvert}^1$
$\vdots \ \ddots \ \vdots$	$\vdots$

(b) *multiclass* (MC) decomposition (only for feature selection)

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

(c) *label powerset* (LP) decomposition

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

$\cdots$

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

(d) *binary relevance* (BR) decomposition

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

Networks in order to find patterns corresponding to unusual dependencies between targets. This Bayesian EMM instance provides MLC with 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 $x = \{a_1, \ldots, a_k, \ell_1, \ldots, \ell_m\}$. 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.

The task of multi-label classification (MLC) is, given a training set $\mathcal{E} \subset \Omega$, to learn a function $f(a_1, \ldots, a_k) \to (\ell_1, \ldots, \ell_m)$ 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) [2,3] approach tackles a multi-label problem by learning a separate classifier $f_i(a_1, \ldots, a_k) \to \ell_i$ for each label ℓ_i, as illustrated in Figure 2d. 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) [17], 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 $\ell_1, \ldots, \ell_{i-1}$. Hence, CC caters for dependencies of labels on multiple other labels, but these dependencies are one-directional: if label ℓ_i depends on the prediction for label ℓ_j, then ℓ_j does not depend on the prediction for ℓ_i.

An alternative approach is *calibrated label ranking* (CLR) [19], 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 [20]. Dependencies shared between larger sets of labels are ignored by CLR.

Finally, a simple way to take label dependencies into account is the *label powerset* (LP) approach [2], treating each combination of labels occuring in the training data as a separate value of a multi-class single-label classification problem (Figure 2c). Hence, LP caters for dependencies between larger sets of labels as they appear in the dataset. However, LP disregards the inclusion lattice that exists between label sets in MLC. If data point x^1 has label set $\{\ell_1, \ell_2\}$, and data point x^2 has label set $\{\ell_1, \ell_2, \ell_3\}$, then the label set for x^1 is a subset of the label set for x^2. However, LP will represent these label sets as unrelated values of a single-label. So even though LP can cater for subtle label dependencies, this inclusion information is not preserved.

We will use each of these techniques for decomposing a multi-label problem into an ordinary classification problem in the third LeGo phase (Section 5).

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 [12];
2. filter out a meaningful subset $S \subseteq P$;
3. use the patterns in S as constructed attributes to enhance multi-label classification methods.

The following sections will explore what we do in each of these phases.

3 Local Pattern Discovery phase

To find the local patterns with which we will enhance the MLC attribute 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 $p\left(a^i_1, \ldots, a^i_k\right) = 1$. 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 two factors: the size of the subgroup, and the degree to which the distribution of the target within the subgroup deviates from the target distribution on the whole dataset. 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, based on model characteristics. 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 [12] 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 [12]. 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 [21] here. Instead, a variant of edit distance for BNs was

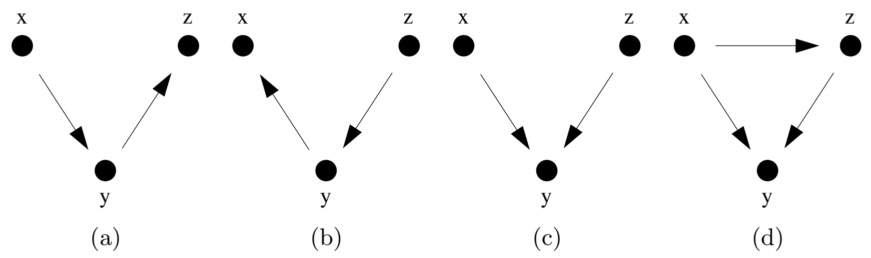

Fig. 3. Example Bayesian networks

introduced, that basically counts the number of violations of the famous theorem by Verma and Pearl on the conditions for equivalence of DAG models [22]:

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.

Definition 1 (Distance between BNs). *Let two Bayesian networks BN_1 and BN_2 be given with the same set of vertices whose size we denote by m. Denote the edge set of their skeletons by S_1 and S_2, and the edge set of their moralized graphs by M_1 and M_2. Let*

$$\ell = \Big|[S_1 \oplus S_2] \;\cup\; [M_1 \oplus M_2]\Big|$$

The distance between BN_1 and BN_2 is defined as:

$$d(BN_1, BN_2) = \frac{2\ell}{m(m-1)}$$

As usual in set theory, $\oplus$ denotes an exclusive disjunction: $X \oplus Y = (X \cup Y) - (X \cap Y)$. The factor $\frac{2}{m(m-1)}$ normalizes the distance to the range $[0, 1]$. The *moralized graph* of a Bayesian network is constructed by drawing the missing edge of every v-structure, i.e. by marrying all unmarried parents.

Notice that this distance only considers the network structures, not the underlying probability distributions. Assigning exceptionality to a subgroup by looking at those distributions had been done before by Leman et al. [11] in a classification setting. As a consequence, our distance does not distinguish between e.g. different signs of correlation: if an edge corresponds to a positive correlation in one network and to a negative correlation in another network, then this edge does not contribute to the distance.

We illustrate the edit distance by computing the mutual distances between the four networks a, b, c, and d shown in Figure 3. We find that $\mathrm{d}(a, b) = 0$

and $\mathrm{d}(a,c) = \mathrm{d}(a,d) = \mathrm{d}(b,c) = \mathrm{d}(b,d) = \mathrm{d}(c,d) = 1/3$. Only the two networks that are equivalent have a distance of 0. If we compare the networks to the independence model i which has no edges at all, we obtain $\mathrm{d}(a,i) = \mathrm{d}(b,i) = 2/3$, and $\mathrm{d}(c,i) = \mathrm{d}(d,i) = 1$.

This distance can now be used to quantify the exceptionality of a pattern; we define the quality measure *edit distance* ($\varphi_{\mathrm{ed}}(p)$) of pattern p to be the distance between the Bayesian network we fit on Ω (BN_Ω) and the Bayesian network we fit on the subgroup corresponding to p (BN_p), i.e. $\varphi_{\mathrm{ed}}(p) = d(BN_\Omega, BN_p)$.

Major changes in structure of the Bayesian networks are easily achieved in a small subset of the data. To counter this effect, the exceptionality measure we employ also contains a component indicating the number of data points the pattern covers. We use the *entropy* ($\varphi_{\mathrm{ent}}(p)$) of the split between the pattern and the rest of the dataset for this, capturing the information content of the split. It favours balanced splits over skewed splits, and again ranges between 0 and 1, with the ends of the range reserved for the extreme cases (pattern being empty or covering the whole dataset, and 50/50 splits, respectively).

Because we do not want to find patterns that have a low quality value on either the edit distance or the entropy measure, we combine them into a weighed measure:

$$\varphi_{\mathrm{weed}}(p) = \sqrt{\varphi_{\mathrm{ent}}(p)} \cdot \varphi_{\mathrm{ed}}(p)$$

The original components ranged from 0 to 1, hence the *Weighed Entropy and Edit Distance* does so too. We take the square root of the entropy, reducing its bias towards 50/50 splits, since we are primarily interested in a subgroup with large edit distance, while mediocre entropy is acceptable.

Notice that the EMM algorithm takes quite some time to complete: for each candidate pattern a Bayesian network is built over the labels, which can be done in $\mathcal{O}\left(m^{2.376}\right)$ time [12]. Since many candidate patterns are considered, this is a high price to pay, even with a relatively small number of labels. However, as stated in Section 2, in the LeGo approach the patterns resulting from the EMM algorithm can be considered partial solutions to local complexities in the data. These solutions do not need to be recomputed every time a classifier is built. Hence, the EMM algorithm needs to be executed only once, and we can afford to invest quite some computation time for this single run. Additionally, the investment needed to compare Bayesian networks as opposed to, for instance, comparing pairwise dependencies between labels, allows us to find anomalies in non-trivial interplay between variables. Such non-trivial modeling has proven its worth in such diverse fields as marine biology [7], traffic accident reconstruction [23], medical expert systems [24], and financial operational risk [25].

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 Pattern Subset Discovery phase

Having outlined the details of local pattern discovery in a multi-label context, we now proceed to the second phase of the LeGo-framework: Pattern Subset Discovery. A common approach for feature subset selection for regular classification problems is to measure some type of correlation between a feature variable and the class variable. A subset of the features S from the whole set P is then determined either by selecting a predetermined number of best features or by selecting all features whose value exceeds a predetermined threshold. Unfortunately this approach is not directly applicable to multi-labeled data without adaptation. We experimented with the following approaches.

A simple way is to convert the multi-label problem into a *multiclass* (MC) classification problem, where each original instance is converted into several new instances, one for each label ℓ_i assigned to the instance, using ℓ_i as the class value (see Figure 2b). However, this transformation does explicitly model label co-occurence for a data point.

An alternative approach is to measure the correlations on the decomposed subproblems produced by the *binary relevance* (BR) decomposition (see Figure 2d). The m different correlation values for each feature are then aggregated. In our experiments, we aggregated with the max operator, i.e., the overall relevancy of a feature was determined by its maximum relevance in one of the training sets of the binary relevance classifiers. The main drawback of this approach is that it treats all labels independently and ignores that a feature might only be relevant for a combination of class labels, but not for the individual labels.

The last approach employs the *label powerset* (LP) transformation (see Figure 2c) in order to also measure the correlation of a feature to the simultaneous absence or occurrence of label sets. Hence, with the dataset transformed into a multiclass problem, common features selection techniques can be applied. The different decomposition approaches are depicted in Figure 2.

After the transformations, we can use common attribute correlation measures for evaluating the importance of an attribute in each of the three approaches. In particular, we used the information gain and the chi-squared statistics value of an attribute with respect to the class variable resulting from the decomposition, as shown in Figures 2b, 2d and 2c. Then we let each of the six feature selection methods select the best patterns from P to form the subset S. The size $|S|$ of the subset is fixed in our experiments (see Section 5).

The approach adapted from multiclass classification of measuring the correlation between each feature and the class variable has known weaknesses such as being susceptible to redundancies within the features. Hence, in order to evaluate the feature selection methods, we will compare them with the baseline method that simply draws S as a random sample from P.

5 Global Modeling phase

For the learning of the global multi-label classification models in the Global Modeling phase, we experiment with several standard approaches including binary

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

(c) combined attributes in the LeGo classifier C_L

Fig. 4. 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)

relevance (BR) and label powerset (LP) decompositions [2,3], as well as a selection of effective recent state-of-the-art learners such as calibrated label ranking (CLR) [19,26], and classifier chains (CC) [17]. The chosen algorithms cover a wide range of different approaches and techniques used for learning multi-label problems (see Section 2.2), and are all included in Mulan, an excellent library for multi-label classification algorithms [2,27].

We combine the multi-label decomposition methods mentioned in Section 5 with several base learners: J48 with default settings [28], standard LibSVM [29], 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 suggested by [29]). 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 [30]. We will refer to LibSVM with the parameter grid search as MetaLibSVM, and denote the used kernel by a superscript *rbf* or *lin*.

For each classifier configuration, we learn three classifiers based on different attribute sets. The first classifier uses only the k attributes that make up the original dataset, and is denoted C_O (Figure 4a). The second classifier, denoted C_S, uses only attributes constructed from our pattern set S. Each of these patterns by definition maps each record in the original dataset to either zero or one. Hence they can be trivially transformed into binary attribute, that together make up the attribute space for classifier C_S (Figure 4b). The third classifier employs both the k original and $|S|$ constructed attributes, in the spirit of LeGo, and is hence denoted C_L (Figure 4c). Its attribute space consists of the k original

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

attributes, and $|S|$ attributes constructed from the pattern set S for a grand total of $k + |S|$ attributes.

6 Experimental setup

To experimentally validate the outlined LeGo method, we will compare the performance of the three classifiers based on different attribute 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.

6.1 Experimental procedure

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.

The *Emotions* dataset [4] consists of 593 songs, from which 8 rhythmic and 64 timbre attributes were extracted. Domain experts assigned the songs to any number of six main emotional clusters: *amazed-surprised*, *happy-pleased*, *relaxing-calm*, *quiet-still*, *sad-lonely*, and *angry-fearful*.

The *Scene* dataset [5] is from the semantic scene classification domain, in which a photo can be classified into one or more of 6 classes. It contains 2407 photos, each of which is divided in 49 blocks using a 7×7 grid. For each block the first two spatial color moments of each band of the LUV color space are computed. This space identifies a color by its lightness (the L* band) and two chromatic valences (the u* and v* band). The photos can have the classes *beach*, *field*, *fall foliage*, *mountain*, *sunset*, and *urban*.

From the biological field we consider the *Yeast* dataset [6]. It consists of micro-array expression data and phylogenetic profiles with 2417 genes of the yeast *Saccharomyces cerevisiae*. Each gene is annotated with any number of 14 functional classes.

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, (if no 10,000 patterns can be found, we report them all), 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 [12]). 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.

Notice that the choice of search lattice traversal does not influence the way the exceptionality measure determines the quality of a pattern. While setting the search depth to 2 does influence the complexity of the pattern in attribute space, this setting is independent from the complexity of the models we fit in label space. Each candidate pattern will be evaluated by fitting a Bayesian network to all its labels, regardless of search parameters such as the search depth.

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 5), i.e. exactly as many pattern-based attributes for C_S and C_L as there are original attributes in C_O.

6.2 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. [19] and [2]). We define $\mathcal{Y}^i = \left\{\ell_j \mid \ell_j^i = 1\right\}$ as the set of assigned labels and $\widehat{\mathcal{Y}}^i$ as the set of predicted labels for a test instance x^i. 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* (Prec) computes the percentage of predicted labels that are relevant, and *Recall* (Rec) computes the percentage of relevant labels that are predicted. Recall and precision allow a commensurate evaluation of an algorithm, in contrast to Hamming loss, which is often used but unfortunately generally favors algorithms with high precision and low recall.

$$\textsc{Prec} = \frac{\sum_i \left|\widehat{\mathcal{Y}}^i \cap \mathcal{Y}^i\right|}{\sum_i \left|\widehat{\mathcal{Y}}^i\right|} \qquad \textsc{Rec} = \frac{\sum_i \left|\widehat{\mathcal{Y}}^i \cap \mathcal{Y}^i\right|}{\sum_i \left|\mathcal{Y}^i\right|}$$

Subset Accuracy (Acc) denotes the percentage of perfectly predicted label sets, basically forming a multi-label version of traditional accuracy.

$$\textsc{Acc} = \frac{\sum_i I\left[\widehat{\mathcal{Y}}^i = \mathcal{Y}^i\right]}{\sum_i 1}\,, \qquad I[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Since the classifiers we consider are able to return rankings on the labels, we also computed the following rank-based loss measures, in which $r(\ell)$ returning the position of label ℓ. *Ranking Loss* (Rank) returns the number of pairs of labels which are not correctly ordered, normalized by the total number of pairs.

$$\textsc{Rank} = \frac{|\{(\ell \in \mathcal{Y}, \ell' \notin \mathcal{Y}) \mid r(\ell) < r(\ell')\}|}{|\mathcal{Y}| \cdot (m - |\mathcal{Y}|)}$$

Average Precision (AvgP) computes the precision at each relevant label in the ranking, and averages these percentages over all relevant labels.

$$\textsc{AvgP} = \frac{1}{|\mathcal{Y}|} \sum_{\ell \in \mathcal{Y}} \frac{|\{\ell' \in \mathcal{Y} \mid r(\ell') \leq r(\ell)\}|}{r(\ell)}$$

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).

6.3 Statistical testing

To draw conclusions from the long list of raw results we obtained, we use the methodology for the comparison of multiple algorithms described by Demšar [31]. First, we perform a Friedman test [32,33] to determine whether the classifiers all perform similarly. This is a non-parametric equivalent of the repeated-measures ANOVA. For each test case we rank the classifiers by their performance. Let T denote the number of test cases, and let r_j be the average rank over all test cases for classifier C_j, where $j \in \{O, S, L\}$. Whenever a tie occurs, average ranks are assigned. The null hypothesis now states that all classifiers perform equivalently and so their average ranks should be equal. Under this null hypothesis, the Friedman statistic

$$\chi_F^2 = \frac{12T}{g(g+1)} \cdot \sum_j \left(r_j - \frac{g+1}{2}\right)^2$$

has a chi-squared distribution with $g-1$ degrees of freedom, when N and k are sufficiently large. Here, g denotes the number of classifiers we are comparing.[3]

[3] notice that while Demšar gives a different equation for the Friedman statistic, this is the equation given by Friedman himself. Equivalence can be shown in four lines of math; the equation shown here is slightly easier to compute, and easier on the eye.

Table 2. Average ranks of different feature selection methods, with critical difference

Selection method	BR		LP		MC		Random	CD
Evaluation method	χ^2	gain	χ^2	gain	χ^2	gain		
Rank	4.445	3.932	3.507	4.263	3.707	4.490	3.657	0.520

If the null hypothesis is rejected, we can determine which classifiers are significantly better than others with a post-hoc test. As proposed by Demšar, we use the Nemenyi test [34], which is similar to the Tukey test for ANOVA. The test entails that the performance of two classifiers is significantly different if the difference between their average ranks is at least the critical difference:

$$CD = q_\alpha \sqrt{\frac{g(g+1)}{6T}}$$

where q_α are critical values based on the Studentized range statistic divided by $\sqrt{2}$. Finally and quite obviously, if the Nemenyi test yields that two classifiers have significantly different performance, the one with the better overall rank performs significantly better.

7 Experimental evaluation

The following subsections are dedicated to different aspects such as as the evaluation of the different pattern subset discovery approaches, the employment of the different attribute sets, the impact of the decomposition approaches, and efficiency.

7.1 Feature selection methods

Before comparing the three classifiers, we take a look at the relative performance of the different feature selection methods. When comparing the performance of the classifier C_L with different feature selection methods over all $T = 300$ test cases, we find the average ranks in Table 2. We compared the Binary Relevance, Label Powerset and MultiClass approach, each with evaluation measures chi-squared and information gain, and the random baseline approach.

The results show that no classifier employing a sophisticated feature selection method significantly[4] outperforms the classifier with random feature selection.

[4] Since we are comparing $g = 7$ different methods here, the critical value with significance level $\alpha = 5\%$ for the chi-squared distribution with $g - 1 = 6$ degrees of freedom equals 12.592. The Friedman statistic for these ranks equals $\chi^2_F = 61.678$, hence the Friedman test is passed. For the Nemenyi test, when comparing 7 methods, the critical value is $q_{0.05} = 2.948$. Hence the critical difference between average ranks becomes $CD = 0.520$.

Conversely, this classifier does significantly outperform several classifiers employing sophisticated feature selection. For the binary relevance and multiclass approaches this is reasonable, since the patterns are explicitly designed to consider interdependencies between labels, while the BR and MC approaches select features based on their correlation with single labels only and hence ignore interdependencies. The label powerset approach should do better in this respect. In fact, the best average rank featured in Table 2 belongs to LP with the chi-squared evaluation measure. Since its improvement over the naive method is not significant, we did not further explore its performance, but that does not mean it is without merit.

Another reason for the bad performance of the feature selection methods is that they evaluate each feature individually. One extreme use case will show the problem: if we replicate each feature n times and we select the n best features according to the presented methods, we will get n times the same (best) feature. In the Local Pattern Mining phase, we produce a high number of additional features, hence we can expect to obtain groups of similar additional features where this problem may appear. The random feature selection does not suffer from this problem. As a result of these experiments, we decided not to use any sophisticated feature selection in the remaining experiments, and focus on the results for random feature selection.

7.2 Evaluation of the LeGo approach

The first row in Table 3 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. We see that both C_O and C_L perform significantly ($\alpha = 5\%$)[5] better than C_S, i.e. the pattern-only classifier cannot compete with the original attributes or the combined classifier. 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. The remainder of Table 3 is concerned with stratified results.

When stratifying the results by base learner (the second block in Table 3), 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 attributes significantly ($\alpha = 5\%$) outperforms the LeGo classifier.

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

[5] Since we are comparing three classifiers, the Friedman statistic equals $\chi^2_F = 52.687$. 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 $\alpha = 5\%$.

Table 3. Comparison of different attribute sets. Average ranks of the three classifiers C_O, C_S, C_L, with critical difference, over all 300 test cases, over all 240 test cases barring J48, over all 60 test cases with a particular base learner, and over all 75 test cases with a particular decomposition method. Bold numbers indicates the top rank in the row, $>$ or $<$ indicate a significant difference to the direct neighbor classifier.

	C_O		C_L		C_S	CD
Overall	1.863	=	**1.797**	>	2.340	0.191
Without J48	1.971	<	**1.733**	>	2.296	0.214
MetaLibSVM$^{\text{rbf}}$	**1.483**	=	1.683	>	2.833	0.428
MetaLibSVM$^{\text{lin}}$	1.900	=	**1.800**	>	2.300	”
LibSVM$^{\text{rbf}}$	2.633	<	**1.683**	=	**1.683**	”
LibSVM$^{\text{lin}}$	1.867	=	**1.767**	>	2.367	”
J48	**1.433**	>	2.050	>	2.517	”
Acc	1.850	=	**1.800**	>	2.350	”
Prec	**1.700**	=	1.883	>	2.417	”
Rec	1.983	=	**1.700**	>	2.317	”
AvgP	1.850	=	**1.833**	>	2.317	0.428
Rank	1.933	=	**1.767**	>	2.300	”
CLR	1.813	=	**1.760**	>	2.427	0.383
LP	**1.773**	=	1.827	>	2.400	”
CC	1.947	=	**1.720**	>	2.333	”
BR	1.920	=	**1.880**	=	2.200	”
Emotions	2.510	<	1.860	=	**1.630**	0.331
Scene	**1.480**	=	1.640	>	2.880	”
Yeast	**1.600**	=	1.890	>	2.510	”

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.

Using only the original attributes seems to be enough for the highly optimized non-linear MetaLibSVM$^{\text{rbf}}$ method, though the difference to the combined attributes is small and not statistically significant. The remaining base learners benefit from the additional local patterns. Notably, when using LibSVM$^{\text{rbf}}$, it is even possible to rely only on the pattern-based attributes in order to outperform the classifiers trained on the original attributes.

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 6.1. The average ranks of the base learners over these 180 test cases can be found in Table 4; Again, the Friedman test is easily passed. The Nemenyi test shows that J48 performs significantly worse than all SVM methods and that

MetaLibSVM$^{\text{rbf}}$ clearly dominates the performance of the SVMs. This last point is not surprising, since the three datasets are known to be difficult and hence not linearly separable [19], which means that an advantage of the RBF-kernel over the linear kernel can be expected. Moreover, the non expensively optimized LibSVM$^{\text{rbf}}$ can be considered to be subsumed by the meta variant since the grid search includes the default settings.

Having just established that J48 is the worst-performing base learner and, additionally, that the similar decision patterns particularly damage the performance of the LeGo classifier, we repeat our overall comparison considering only the SVM variants. Moreover, SVMs are conceptually different from decision tree learners, which additionally justifies the separate comparison. The average ranks of the three classifiers C_O, C_S, and C_L on the remaining 240 test cases can be found in the second row of Table 3. This time, the Nemenyi test yields that on the SVM methods the LeGo classifier is generally significantly better than the classifier built from original attributes, even though for MetaLibSVM$^{\text{rbf}}$ this is not the case.

When stratifying the results by quality measure (the third block in Table 3) we find that the results are consistent over the chosen measures. For all measures we find that C_L significantly outperforms C_S, and C_O always outperforms C_S though not always significantly. Additionally, for all measures except precision, C_L outranks C_O, albeit non-significantly. This consistency provides evidence for robustness of the LeGo method.

The fourth block in Table 3 concerns the results stratified by transformation technique. With the exception of the label powerset approach, which by itself respects relatively complex label dependencies, all approaches benefit from the combination with the constructed LeGo attributes, though the differences are not statistically significant. Of peculiar interest is the benefit for the binary relevance approach, which in its original form considers each label independently. Though the Friedman test is not passed, the trend is confirmed by the results of CC, which additionally include attributes from the preceding base classifiers' predictions.

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 an attribute enriching approach, adding an attribute set C. The result comparing the performance of the different attribute sets arrives at $O \cup S \cup C$ (rank 2.84) followed by $O \cup S$ (3.34), $O \cup C$ (3.53), O (3.6), S (3.89) and $S \cup C$ (3.97) (significant difference only between first and both last combinations). Hence, adding C has a similar effect on performance than adding S, and BR particularly benefits if both are added, which demonstrates that the patterns based on local exceptionalities provide additional information on the label dependencies which is not covered by C.

In the last block of Table 3 we see that results vary wildly when stratified by dataset. We see no immediate reason why this should be the case; perhaps a study involving more datasets could be fruitful in this respect.

Table 4. 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

Table 5. Comparison of the decomposition approaches. The first block compares the approaches for all base learner combinations, the second one restricts on the usage of MetaLibSVM$^{\text{rbf}}$. The first row in each block indicates the average ranks with respect to all evaluation metrics, whereas the following rows distinguish between the individual measures.

Measure	CLR		LP		CC		BR	CD
all & all BC	**1.909**	$>$	2.462	$=$	2.700	$=$	2.929	0.313
Acc	3.400	$<$	**1.489**	$=$	1.722	$>$	3.389	0.700
Prec	1.989	$>$	3.467	$=$	3.111	$<$	**1.433**	”
Rec	2.156	$=$	**1.956**	$=$	2.422	$>$	3.467	”
AvgP	**1.000**	$>$	2.778	$=$	3.111	$=$	3.111	”
Rank	**1.000**	$>$	2.622	$=$	3.133	$=$	3.244	”
all & MetaLibSVM$^{\text{rbf}}$	2.111	$=$	**1.911**	$>$	2.911	$=$	3.067	0.700
Acc	3.667	$<$	**1.000**	$=$	2.000	$=$	3.333	1.563
Prec	2.111	$=$	3.444	$=$	3.444	$<$	**1.000**	”
Rec	2.778	$<$	**1.000**	$=$	2.333	$=$	3.889	”
AvgP	**1.000**	$=$	2.111	$=$	3.444	$=$	3.444	”
Rank	**1.000**	$=$	2.000	$=$	3.333	$=$	3.667	”

7.3 Evaluation of the decompositive approaches

We can learn more from Table 3 when more blocks are additionally stratified by evaluation measure. Indeed, the decision of the feature base does not seem to have a differing impact on the metrics (for the SVM learners, not shown in the table). The only exception appears to be micro-averaged precision, for which C_O yields a small advantage over C_L. But as Table 5 demonstrates, the situation changes with respect to the decompositive approach used. As we can see in the upper block, there are clear tendencies regarding the preference for a particular metric.

For instance, LP has a clear advantage in terms of subset accuracy, which only CC is able to approximate. This is natural, since both approaches are dedicated to the prediction of a correct labelset. In fact, LP only predicts labelsets previously seen in the training data. CC behaves similarly, as is shown in the following: if we consider only the additional attributes from the previous predictions (i.e. attribute set C), then we found that CC behaves similar to a sequence tagger. Hence, for a particular sequence of labels $\ell_1, \ldots, \ell_{i-1}$ the i-th classifier in the chain will tend to predict $\ell_i = 1$ (or $\ell_i = 0$ respectively) only if $\ell_1, \ldots, \ell_i$ existed in the training data. The advantage of LP and CC is confirmed in the bottom block, which restricts the comparison to the usage of the most accurate base learner MetaLibSVM$^{\mathrm{rbf}}$.

Precision is dominated by BR, followed by CLR. This result is obtained by being very cautious at prediction, as the values for recall show. Especially the highly optimized SVM is apparently fitted towards predicting a label only if it is very confident that the estimation is correct. It is not clear whether this is due to the high imbalance of the binary subproblems, e.g. compared to pairwise decomposition. CLR shows to be more robust, though a bias towards underestimating the size of the label sets is visible. Especially in this case the bias may originate from the conservative BR classifiers, which are included in the calibrated ensemble, since the difference between precision and recall is clearly higher for MetaLibSVM$^{\mathrm{rbf}}$.

A contrary behavior to BR is shown by LP, which dominates recall, especially for MetaLibSVM$^{\mathrm{rbf}}$, but completely neglects precision. This indicates a preference for predicting the more rare large labelsets. The best balance between precision and recall is shown by CLR. Even for MetaLibSVM$^{\mathrm{rbf}}$, for which the underestimation leads to low recall, but for which the competing classifier chains obtain the worst precision values together with LP.

The good balancing properties of CLR are confirmed by the results on the ranking losses, which are clearly dominated by CLR's good ability to produce a high density of relevant labels at the top of the the rankings. The high recall of LP corresponds with good ranking losses, but the low ranks of BR demonstrate that its high precision is not due to a good ranking ability. This behavior was already observed e.g. in [19] and [20] where BR often correctly pushed a relevant class at the top, but obtained poor ranking losses. Similarly, CC's base classifiers are trained independently without a common basis for confidence scores and hence achieve a low ranking quality.

If we give the same weight to the five selected measures, we observe that CLR significantly outperforms the second placed LP if all base learner are considered, and slightly loses against LP if MetaLibSVM$^{\mathrm{rbf}}$ is used (top row in both blocks in Table 3).

7.4 Efficiency

Apart from the unsatisfactory performance of the J48 approach compared to SVM approaches, Table 4 also indicates that compared to the standard LibSVM approach, the extra computation time invested in the MetaLibSVM parameter grid search is rewarded with a significant increase in classifier performance. For both the linear and the RBF kernel we see that the MetaLibSVM approach outperforms the LibSVM approach, although this difference is only significant for the RBF kernel. A more exhaustive parameter-optimizing search will probably be beneficial, since the grid search considers arbitrary parameter values. Whether the performance increase is worth the invested time is a question of preference. In the case where time and computing power are not limited resources, the increased performance is clearly worthwhile.

From a practical point of view, it is also interesting to analyze the efficiency of using the original attributes in comparison to using the constructed attributes. We expected an improvement in complexity just from the fact that the pattern attributes are binary in contrast to the more complex nature of the original attributes in the used datasets. In addition, it is well known that the possibly resulting sparseness of the binary attributes may also boost algorithms like SVMs.

For the comparison between training of C_O and C_S we focus on the Binary Relevance decomposition setting in order to allow a balanced comparison over the three datasets, since the complexity of BR scales linearly with respect to the number of classes. For J48 as base learner, we observed a reduction of training costs from 28% (*Emotions*) over 31% (*Scene*) to 47% for *Yeast*. For LibSVM$^{\mathrm{rbf}}$, the difference was more pronounced, with a reduction of 60%, 52% and 50%, resp. We obtained a similar picture for LibSVM$^{\mathrm{lin}}$, with a reduction of even 82% (*Emotions*) and 65% (*Scene*), except for *Yeast*, where training C_S surprisingly takes almost 7 times longer than training C_O. This case is very likely an exception since comparing LibSVM$^{\mathrm{lin}}$ and hence using different parameters shows again a clear reduction. The numbers for MetaLibSVM$^{\mathrm{lin}}$ and MetaLibSVM$^{\mathrm{rbf}}$ are omitted since they show a similar picture but are more difficult to compare directly since they always also include testing time.

Note that training C_L of course takes more computation time since we employ both attribute sets, but the overhead of using the constructed attributes from the patterns is clearly relatively small. Also, notice that the overhead needs to be invested only once for training the classifier, possibly off-line, and that the resulting trained classifier can then be used again and again for classifying data; if one wants to classify more than once, the added complexity diminishes.

8 Discussion and related work

In traditional subgroup discovery, it is common to traverse the search space exhaustively [35, 36]. Usually the assumption is made that all attributes of the dataset are nominal. Most current work on subgroup discovery still incorporates this restriction [37], but recently subgroup discovery on numeric domains has received more attention [38]. In this work the search space is traversed exhaustively even though it contains numeric attributes. However, this can only be done under the restriction that the employed quality measure has a certain anti-monotonic property: if one could compute an optimistic estimate for the quality of any refinement of a given candidate pattern, the search space can be significantly pruned.

Conversely, we choose to let our work be free from the restrictions of nominality of the datasets and anti-monotonicity of the quality measure, since many real-life datasets contain numeric attributes and we expect to benefit from using a more complex quality measure. We also expect little disadvantage from using heuristic rather than exhaustive search in the Local Pattern Discovery phase. The found patterns are afterwards put through the Pattern Subset Discovery phase, where they are subjected to feature selection which is heuristic by definition, hence there is really no point in enforcing an exhaustive search in the Local Pattern Discovery phase.

The applied Local Pattern Discovery algorithm was created to find patterns that are interesting by themselves. The output of the algorithm is therefore not specifically tailored to be useful in a classification setting; this is not a guiding principle in the Exceptional Model Mining process. To the best of our knowledge, this work is a first shot at testing the utility for classification of the result of such a stand-alone multi-label pattern discovery process. Some recent sophisticated classifiers, for instance the multi-label lazy associative classifiers [39], are also based on local patterns. However, these patterns serve only the classifier, hence the different phases, as present in the LeGo framework, are not as separated as they are in our work. Similarly, Cheng and Hüllermeier [15] incorporate additional attributes that encode the label distribution in the direct neighborhood by, in effect, stacking the output of a k-Nearest Neighbor classifier. However, this has to be done at (training and testing) runtime and cannot be done separately and beforehand.

Other known stacking approaches include the outcome of global classifiers. Godbole and Sarawagi [40] use the outputs of a BR-SVM classifier as additional input attributes for second-level SVMs. Similarly, Tsoumakas et al. [41] replace all original attributes by the predicted scores of a BR. The scores are additionally filtered according to their correlation to each other. The employed classifier chains [17] rely on stacking the outcomes of the predetermined sequence of previous binary relevance classifiers, which permits to model conditional dependencies, but it does not rely on locality. Zhang and Zhang [18] also try to model label dependencies and start from the premise of eliminating the conditional dependency between the input attributes $a_1, \ldots, a_k$ and the individual labels by computing the errors e_i as difference between true label ℓ_i and the prediction. The isolated dependencies between labels are then approximated by the result of building a Bayesian network on these errors. A new BR classifier is then learned for each class with the set of alleged parents as additional attributes. The very recent LIFT algorithm selects particularly representative centroids in the positive and negative examples of a class by k-means clustering and then replaces the original attributes of an instance by the distances to these representatives [42]. One may also interpret this approach as a different, pragmatic way of computing new suitable principal components and hence dimensionality reduction, which apparently works quite well.

9 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 attribute 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 attributes. The J48 classifier does not benefit from the local pattern addition, which can be attributed 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 attributes generally cannot *replace* the original attributes without significant loss in classification performance. We find this reasonable, since these attributes 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 attributes.

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 [12], 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 attributes [43] 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 [44]. 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 independent. 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 Foundation (DFG). We also gratefully acknowledge support by the Frankfurt Center for Scientific Computing.

References

1. W. Duivesteijn, E. Loza Mencía, J. Fürnkranz, A. Knobbe, Multi-label LeGo — Enhancing Multi-label Classifiers with Local Patterns, Advances in Intelligent Data Analysis XI – 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012, Proceedings. LNAI 7619, pp. 114-125, 2012. `http://www.ke.tu-darmstadt.de/publications/papers/IDA-12.pdf`.
2. G. Tsoumakas, I. Katakis, I. P. Vlahavas, Mining Multi-label Data, Data Mining and Knowledge Discovery Handbook, Springer, pp. 667–685, 2010.
3. G. Tsoumakas, I. Katakis, Multi-Label Classification: An Overview, International Journal of Data Warehousing and Mining 3 (3), pp. 1–13, 2007.
4. K. Trohidis, G. Tsoumakas, G. Kalliris, I. P. Vlahavas, Multi-Label Classification of Music into Emotions, Proc. 9th International Conference on Music Information Retrieval, pp. 325–330, 2008.
5. M. R. Boutell, J. Luo, X. Shen, C. M. Brown, Learning Multi-Label Scene Classification, Pattern Recognition 37 (9), pp. 1757–1771, 2004.
6. A. Elisseeff, J. Weston, A Kernel Method for Multi-Labelled Classification, Advances in Neural Information Processing Systems 14, pp. 681–687, MIT Press, Cambridge MA, 2002.
7. R. T. Paine, Food Web Complexity and Species Diversity, The American Naturalist 100 (910), pp. 65–75, 1966.
8. J. Fürnkranz, A. Knobbe (eds.), Special Issue: Global Modeling Using Local Patterns, Data Mining and Knowledge Discovery journal 20 (1), 2010.
9. A. Knobbe, B. Crémilleux, J. Fürnkranz, M. Scholz, From local patterns to global models: The LeGo approach to data mining, Proc. ECMLPKDD'08 WS From Local Patterns to Global Models, pp. 1–16, 2008.
10. M. van Leeuwen, Maximal Exceptions with Minimal Descriptions, Data Mining and Knowledge Discovery journal, special issue ECMLPKDD'10, 21 (2), pp. 259–276, Springer, 2010.
11. D. Leman, A. Feelders, A. Knobbe, Exceptional Model Mining, Proc. ECML/PKDD (2), pp. 1–16, 2008.
12. 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.
13. A. Knobbe, J. Valkonet, Building Classifiers from Pattern Teams, From Local Patterns to Global Models: Proc. ECML PKDD 2009 Workshop, Slovenia, 2009.
14. 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.
15. W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Machine Learning 76 (2-3), pp. 211–225, 2009.

16. S.-H. Park, J. Fürnkranz, Multi-Label Classification with Label Constraints, Proc. ECML/PKDD-08 Workshop on Preference Learning (PL-08), pp. 157–171, 2008.
17. J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier Chains for Multi-label Classification, Proc. ECML PKDD, pp. 254–269, Springer, 2009.
18. M.-L. Zhang, K. Zhang, Multi-label learning by exploiting label dependency, Proc. KDD 2010, pp. 999–1008, 2010.
19. J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, K. Brinker, Multilabel Classification via Calibrated Label Ranking, Machine Learning 73 (2), pp. 133–153, 2008.
20. E. Loza Mencía, J. Fürnkranz, Efficient Pairwise Multilabel Classification for Large-Scale Problems in the Legal Domain, Proc. ECML/PKDD (2), pp. 50–65, Springer-Verlag, 2008.
21. L. G. Shapiro, R. M. Haralick, A Metric for Comparing Relational Descriptions, IEEE Trans. Pattern Anal. Mach. Intell. 7, pp. 90–94, 1985.
22. T. Verma, J. Pearl, Equivalence and Synthesis of Causal Models, Proc. UAI, pp. 255–270, 1990.
23. G. A. Davis, Bayesian reconstruction of traffic accidents, Law, Probability and Risk 2, pp. 69–89, 2003.
24. F. J. Díez, J. Mira, E. Iturralde, S. Zubillaga, DIAVAL, a Bayesian expert system for echocardiography, Artificial Intelligence in Medicine 10, pp. 59–73, 1997.
25. M. Neil, N. Fenton, M. Tailor, Using Bayesian Networks to Model Expected and Unexpected Operational Losses, Risk Analysis 25 (4), 2005.
26. G. Tsoumakas, E. Loza Mencía, I. Katakis, S.-H. Park, J. Fürnkranz, On the Combination of Two Decompositive Multi-Label Classification Methods, Proc. ECML/PKDD-09 Workshop on Preference Learning (PL-09), pp. 114–129, 2009.
27. G. Tsoumakas, J. Vilcek, E. Spyromitros Xioufis, I. P. Vlahavas, Mulan: A Java Library for Multi-Label Learning, Journal of Machine Learning Research 12, pp. 2411–2414, 2011.
28. I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, San Francisco, 2005.
29. C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
30. 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.
31. J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7, pp. 1–30, 2006.
32. M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32, pp. 675–701, 1937.
33. M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11, pp. 86–92, 1940.
34. P. B. Nemenyi, Distribution-free multiple comparisons, PhD thesis, Princeton University, 1963.
35. J. Friedman, N. Fisher, Bump-Hunting in High-Dimensional Data, Statistics and Computing 9(2), pp. 123–143, 1999.
36. W. Klösgen, Subgroup Discovery, Handbook of Data Mining and Knowledge Discovery, ch. 16.3, Oxford University Press, New York, 2002.
37. H. Grosskreutz, S. Rüping, S. Wrobel, Tight Optimistic Estimates for Fast Subgroup Discovery, Proc. ECML/PKDD (1), pp. 440–456, 2008.
38. H. Grosskreutz, S. Rüping, On Subgroup Discovery in Numerical Domains, Data Mining and Knowledge Discovery 19(2), pp. 210–226, 2009.

39. A. Veloso, W. Meira Jr., M. A. Gonçalves, M. J. Zaki, Multi-label Lazy Associative Classification, Proc. PKDD, pp. 605–612, 2007.
40. S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference (PAKDD 2004), Sydney, Australia, May 26-28, 2004, pp 22–30, 2004.
41. G. Tsoumakas, A. Dimou, E. Spyromitros Xioufis, V. Mezaris, I. Kompatsiaris, I. Vlahavas, Correlation based pruning of stacked binary relevance models for multi-label learning, Proceedings of the 1st International Workshop on Learning from Multi-Label Data (MLD'09), pp. 101–116, 2009.
42. M.-L. Zhang, Lift: Multi-label learning with label-specific features, Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 1609–1614, 2011.
43. M. van Leeuwen, A. J. Knobbe, Non-Redundant Subgroup Discovery in Large and Complex Data, Proc. ECML PKDD (3), pp. 459–474, 2011.
44. K. Sechidis, G. Tsoumakas, I. P. Vlahavas, On the Stratification of Multi-label Data, Proc. ECML PKDD (3), pp. 145–158, 2011.