Exceptional Model Mining

Duivesteijn, W.

Chapter 9

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

Contrary to ordinary classification, in multi-label classification (MLC) one can assign more than one class label to each record [106, 107]. For instance, when we have the earth's continents as classes, a news article about the March 2013 election of Pope Francis, who was born in Argentina, could be labeled with the *Europe* and *South 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 semantic scene classification [6], protein function classification [28], and music categorization [103].

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, suggesting 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, as we have for instance seen in the *Pisaster* example of Chapter 2, where the dependence between *Haliclona* and *Anisodoris* is conditional on the

presence of *Pisaster*. The ability to consider such interplay is an essential element of good multi-label classifiers.

In this chapter we investigate incorporating locally exceptional interactions between labels in MLC, as an instance of the *LeGo framework* [37, 57]. In this framework, the KDD process is split up into several phases. First, local models are found, each representing only part of the data. Then, a subset of these models is selected. Finally, this subset is employed in constructing a global model. The crux is that straightforward classification methods can be used for building a global classifier, if the locally exceptional interactions between labels are represented by attributes constructed from descriptions found in the local modeling phase.

The descriptions representing these locally exceptional interactions are found with the EMM instance from Chapter 6: modeling the conditional dependencies between the labels by a Bayesian network, and striving to find descriptions for which the learned network has a substantially different structure than the network learned on the whole dataset. These descriptions can each be represented by a binary attribute of the data. At the end of this chapter we demonstrate that the integration of these description-based attributes into the classification process improves classifier performance. We also investigate whether the newly generated binary attributes are expressive enough to replace the original descriptive attributes, while maintaining classifier performance and increasing efficiency.

9.1 The LeGo Framework

As mentioned, the work in this chapter relies heavily on the LeGo framework [37, 57]. This framework assumes that the induction process is not executed by running a single learning algorithm, but rather consists of consecutive phases, as illustrated in Figure 9.1. In the first phase, a Local Pattern Mining algorithm is employed in order to obtain a number of informative descriptions, which can serve as attributes to be used in the subsequent phases. These descriptions can be considered partial solutions to local complexities in the data. In the second and third phase, the descriptions are filtered to reduce redundancy, and the selected descriptions are combined in a final global model, which is the outcome of the process.

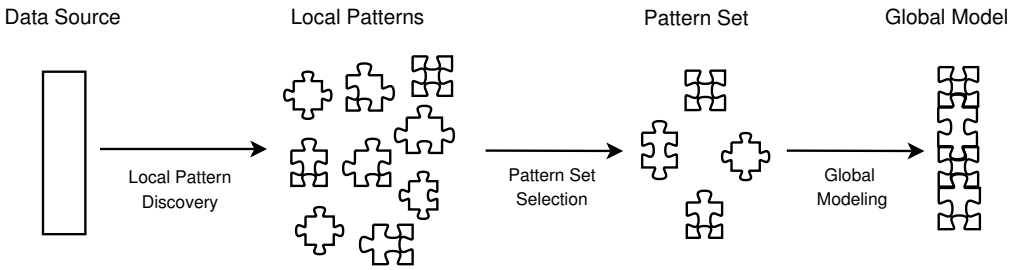

Figure 9.1: The LeGo framework.

The idea of the LeGo framework is that, instead of attacking the induction task in the original representation, we transform it by an automated process to a representation that already resolves a number of complexities in the original task. Then, the new representation can be approached with a standard Global Modeling technique, such as Support Vector Machines (SVMs) with linear kernels, since the potentially hard aspects of the original representation have been accounted for in the new representation by the descriptions. Generally, for this automated process, any of the existing Local Pattern Mining algorithms can be employed, thus benefiting from the wealth of LPM algorithms that has grown over the last decade.

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 Mining 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. By contrast, in most Local Pattern Mining methods, 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 descriptions) as partial solutions to local complexities in the data. The descriptions, which can be interpreted as new virtual attributes, 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 [37], particularly regular binary classification [59, 100], but we have reasons for choosing

this approach in the context of multi-label classification (MLC). It is often mentioned that in MLC, one needs to consider potential interactions between the labels, and that simultaneous classification of the labels may benefit from knowledge about such interactions [11, 87, 92, 118].

In the remainder of this chapter, we will identify the targets in EMM with the labels in MLC, hence the terms *the labels* and *the targets* refer to the exact same thing. Similarly, to adhere to the usual terminology in classification, we will let *the attributes* refer to any attribute of the dataset which can be used by the classifier to define its decision boundary on. Hence, an attribute can both be a descriptive attribute from the original dataset, or a constructed attribute built from a description found in the Local Pattern Mining phase. Finally, since we employ commonly known feature selection methods in the Pattern Subset Discovery phase, we will occasionally refer to attributes as *features*. Neither the term "attribute" nor the term "feature" can ever refer to a target/label.

9.2 Multi-label Classification

The task of Multi-Label Classification (MLC) is, given a training set $\mathcal{E} \subseteq \Omega$, to learn a function $f : (a_1, \ldots, a_k) \rightarrow (\ell_1, \ldots, \ell_m)$ which predicts the labels for a given record. Many multi-label learning techniques reduce this problem to ordinary classification, where for a given record exactly one *class* is predicted, rather than multiple labels. We will use each of these techniques for decomposing a multi-label problem into an ordinary classification problem in the third LeGo phase (cf. Section 9.3.3).

The widely used *binary relevance* (BR) [106, 107] 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 9.2b. At query time, each binary classifier predicts whether its class is relevant for the query record or not, producing a set of relevant labels. Obviously, BR ignores interdependencies between classes since it learns the relevance of each class independently.

One could address this problem with *classifier chains* (CC) [92], which can model label dependencies since they stack the model outputs: the prediction of the model for label ℓ_i depends on the predictions for labels

Attributes			Labels $\in \{0,1\}^m$		
a_1^1,	$\ldots$,	a_k^1	ℓ_1^1,	$\ldots$,	ℓ_m^1
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
a_1^N,	$\ldots$,	a_k^N	ℓ_1^N,	$\ldots$,	ℓ_m^N

(a) Input training set.

Attributes			$\text{Class}_1 \in \{0,1\}$
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			$\text{Class}_m \in \{0,1\}$
a_1^1,	$\ldots$,	a_k^1	ℓ_m^1
$\vdots$	$\ddots$	$\vdots$	$\vdots$
a_1^N,	$\ldots$,	a_k^N	ℓ_m^N

(b) *Binary Relevance* (BR) decomposition.

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	$y_{\lvert\mathcal{Y}^1\rvert}^1$
$\vdots$	$\ddots$	$\vdots$	$\vdots$

(c) *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	$\mathcal{Y}^N$

(d) *Label Powerset* (LP) decomposition.

Figure 9.2: Decomposition of multi-label training sets into binary (BR) or multiclass problems (MC, LP). Here, $\mathcal{Y}^i = \{y_1^i, \ldots, y_{|\mathcal{Y}^i|}^i \mid y_j^i \in \mathcal{L}\}$ denotes the assigned labels $\{\ell_j \mid \ell_j^i = 1\}$ to record r^i. MC replicates each record $|\mathcal{Y}^i|$ times, once with each assigned label. In LP, the predicted label set is from the set $\{\mathcal{Y}^i \mid i = 1, \ldots, m\} \subseteq 2^{\mathcal{L}}$ of label sets seen in the training data.

$\ell_1, \ldots, \ell_{i-1}$. Hence, CC captures dependencies of labels on multiple other labels, but the 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) [36], where the key idea is to learn one classifier for each binary comparison of labels. CLR learns binary classifiers $f_{ij} : (a_1, \ldots, a_k) \to (\ell_i \succ \ell_j)$, 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. The decomposition into pairs of classes has the advantage of simpler subproblems and hence commonly more accurately performing models [74]. CLR ignores dependencies shared between larger sets of labels.

Finally, a simple way to take label dependencies into account is the *label powerset* (LP) approach [107], treating each combination of labels occurring in the training data as a separate value of a multi-class single-label classification problem (Figure 9.2d). 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 record r^1 has label set $\{\ell_1, \ell_2\}$, and record r^2 has label set $\{\ell_1, \ell_2, \ell_3\}$, then the label set for r^1 is a subset of the label set for r^2. However, LP will represent these label sets as unrelated values of a single class. So while LP captures subtle label dependencies, this inclusion information is not preserved.

9.3 LeGo Components

As Figure 9.1 illustrates, there are three main components in the LeGo framework. In the following three subsections we will outline what we do in each of these steps.

9.3.1 Local Pattern Mining Phase

In the first phase of the LeGo framework, Local Pattern Mining, we use the Exceptional Model Mining instance defined in Chapter 6. With quality measure φ_{weed}, we find a set P of descriptions for which a Bayesian network, modeling the conditional dependence relations between our labels $\ell_1, \ldots, \ell_m$, has an unusual structure.

9.3.2 Pattern Subset Discovery Phase

Having positioned Local Pattern Mining 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 classification problems is to measure some type of correlation between an attribute and the label. A subset of the attributes S from the whole set P is then determined either by selecting a number of best attributes or by selecting all attributes whose value exceeds a threshold.

Each description from the set P we found in the previous LeGo phase, is by definition a function (cf. Section 2.1), mapping the descriptive attributes of a record in the original dataset to either zero or one. Hence, a description can be trivially transformed into a binary attribute of the dataset, detailing for each record whether it is covered by the description or not. This representation as a binary attribute enables determining the correlation between an element from P and a single class label. However, in MLC, multiple class labels are available, leading to multiple correlation assessments for an element from P. Depending on the effect one strives to achieve, these assessments can be combined in a selection criterion in multiple ways. 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 record is converted into several new records, one for each label ℓ_i assigned to the record, using ℓ_i as the class value (see Figure 9.2c). However, this transformation does explicitly model label co-occurrence for a record, not taking the underlying label decomposition into account.

An alternative approach is to measure the correlations on the decomposed subproblems produced by the *binary relevance* (BR) decomposition (see Figure 9.2b). The m different correlation values for each attribute are then aggregated. In our experiments, we aggregated with the max operator, i.e., the overall relevancy of an attribute 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 an attribute 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 9.2d) in order to also measure the correlation of an attribute 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 9.2.

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 χ^2 statis-

tic of an attribute with respect to the class variable resulting from the decomposition, as shown in Figures 9.2b, 9.2c and 9.2d. Then we let each of the six feature selection methods select the best descriptions from P to form the subset S. The size |S| of the subset is fixed in our experiments (see Section 9.3.3).

The approach, adapted from multiclass classification, to measure the correlation between each attribute and the class variable has known weaknesses such as being susceptible to redundancies within the attributes. 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.

9.3.3 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 relevance (BR) and label powerset (LP) decompositions [106, 107], as well as a selection of effective recent state-of-the-art learners such as calibrated label ranking (CLR) [36, 105], and classifier chains (CC) [92]. The chosen algorithms cover a wide range of approaches and techniques used for learning multi-label problems (see Section 9.2), and are all included in Mulan, a library for multi-label classification algorithms [107, 108].

We combine the multi-label decomposition methods mentioned in Section 9.3.3 with several base learners: J48 with default settings [113], standard LibSVM [10], 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 [10]). 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 [29]. 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 (cf. Figure 9.3a). The second classifier, denoted C_S, uses only attributes constructed from our description

Attributes			Labels		
a_1^1,	...,	a_k^1	ℓ_1^1,	...,	ℓ_m^1
a_1^2,	...,	a_k^2	ℓ_1^2,	...,	ℓ_m^2
⋮	⋱	⋮	⋮	⋱	⋮
a_1^N,	...,	a_k^N	ℓ_1^N,	...,	ℓ_m^N

(a) Input training set C_O.

Attributes			Labels		
$D_1(a_1^1,\ldots,a_k^1)$,	...,	$D_{\lvert S\rvert}(a_1^1,\ldots,a_k^1)$	ℓ_1^1,	...,	ℓ_m^1
$D_1(a_1^2,\ldots,a_k^2)$,	...,	$D_{\lvert S\rvert}(a_1^2,\ldots,a_k^2)$	ℓ_1^2,	...,	ℓ_m^2
⋮	⋱	⋮	⋮	⋱	⋮
$D_1(a_1^N,\ldots,a_k^N)$,	...,	$D_{\lvert S\rvert}(a_1^N,\ldots,a_k^N)$	ℓ_1^N,	...,	ℓ_m^N

(b) Transformation into description space C_S.

Attributes						Labels		
a_1^1,	...,	a_k^1,	$D_1(a_1^1,\ldots,a_k^1)$,	...,	$D_{\lvert S\rvert}(a_1^1,\ldots,a_k^1)$	ℓ_1^1,	...,	ℓ_m^1
a_1^2,	...,	a_k^2,	$D_1(a_1^2,\ldots,a_k^2)$,	...,	$D_{\lvert S\rvert}(a_1^2,\ldots,a_k^2)$	ℓ_1^2,	...,	ℓ_m^2
⋮	⋱	⋮	⋮	⋱	⋮	⋮	⋱	⋮
a_1^N,	...,	a_k^N,	$D_1(a_1^N,\ldots,a_k^N)$,	...,	$D_{\lvert S\rvert}(a_1^N,\ldots,a_k^N)$	ℓ_1^N,	...,	ℓ_m^N

(c) Combined attributes in the LeGo classifier C_L.

Figure 9.3: A multi-label classification problem (a), its representation in description space (b) given the set of descriptions $D_1,\ldots,D_{|S|}$, and the LeGo combination (c).

set S (cf. Figure 9.3b). The third classifier employs both the k original and $|S|$ constructed attributes, in the spirit of LeGo, and is hence denoted C_L (cf. Figure 9.3c). Its attribute space consists of the k original attributes, and $|S|$ attributes constructed from the description set S for a grand total of $k + |S|$ attributes.

9.4 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 feature selection methods, and of the classification approaches. All experiments

are performed on the *Emotions*, *Scene*, and *Yeast* datasets introduced in Chapter 6; see Table 6.1 for statistics regarding the datasets.

All statistics on the classification processes are estimated via 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 Mining and Pattern Subset Discovery phase separately for each fold.

For every fold on every dataset, we determine the best $10,000$ descriptions, (if no $10,000$ descriptions can be found, we report them all), measuring the exceptionality with φ_{weed} as described in Chapter 6. We configure the beam search algorithm for EMM with a width of $w = 10$ and a depth of $d = 2$. The modest search depth is selected deliberately, to prevent producing an abundance of highly similar descriptions. We further bound the search by setting the minimal coverage of a description 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 select $|S| = k$ descriptions (cf. Section 9.3.3) from the generated set P, i.e. exactly as many description-based attributes for C_S and C_L as there are original descriptive attributes in C_O.

9.4.1 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 cf. [36] and [107]). We define $\mathcal{Y}^i = \{\ell_j \mid \ell_j^i = 1\}$ as the set of assigned labels and $\widehat{\mathcal{Y}}^i$ as the set of predicted labels for a test record r^i. We consider these 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 records, *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. We have

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

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

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

Since the classifiers we consider are able to return rankings on the labels, we also compute the following rank-based loss measures, in which $\text{rank}(\ell)$ returns 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, i.e.

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

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

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

These two ranking measures are computed for each record and then averaged over all records.

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).

9.4.2 Statistical Testing

To draw conclusions from the long list of raw results we obtained, we use again the methodology for the comparison of multiple algorithms described

by Demšar [19], as introduced in Section 8.2.3. Instead of comparing g quality measures, here we compare the three classifiers C_O, C_S, and C_L. Again, if the difference between the average ranks of two classifiers surpasses the computed critical difference, the better-ranked classifier performs significantly better.

9.5 Experimental Evaluation

The following subsections are dedicated to different aspects such 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.

9.5.1 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 9.1. We compared the Binary Relevance, Label Powerset and MultiClass approach, each with evaluation measures χ^2 and information gain, and the random baseline approach.

The results show that no classifier employing a sophisticated feature selection method significantly[1] outperforms the classifier with random feature selection. Conversely, the classifier with random feature selection *does* significantly outperform several classifiers employing sophisticated feature selection. For the binary relevance and multiclass approaches this is reasonable, since the descriptions are explicitly designed to consider interdependencies between labels, while the BR and MC approaches select attributes

[1] 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 $\chi_F^2 = 61.678$, 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 $CD = 0.520$.

Table 9.1: Average ranks of the feature selection methods (cf. Section 9.3.2), with critical difference. The results for random feature selection are not split out over the two attribute correlation measures, since none is used.

	BR		LP		MC		Random	CD
	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

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 9.1 belongs to LP with the χ^2 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 attribute individually. One extreme use case will illustrate the problem: if we replicate each attribute x times and we select the x best attributes according to the presented methods, we will get x times the same attribute. In the Local Pattern Mining phase, we produce a high number of candidate attributes, hence we can expect to obtain groups of similar candidates. The random feature selection does not suffer from this problem. Hence, for the subsequent experiments, we decided not to use any sophisticated feature selection in the remaining experiments, and focus on the results for random feature selection.

9.5.2 Evaluation of the LeGo Approach

The first row in Table 9.2 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. We see that both C_O and C_L perform significantly ($\alpha = 5\%$)[2] better than

[2]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 $\alpha = 5\%$.

Table 9.2: 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^{rbf}	**1.483**	=	1.683	>	2.833	0.428
MetaLibSVM^{lin}	1.900	=	**1.800**	>	2.300	"
LibSVM^{rbf}	2.633	<	**1.683**	=	**1.683**	"
LibSVM^{lin}	1.867	=	**1.767**	>	2.367	"
J48	**1.433**	>	2.050	>	2.517	"
ACC	1.850	=	**1.800**	>	2.350	0.428
PREC	**1.700**	=	1.883	>	2.417	"
REC	1.983	=	**1.700**	>	2.317	"
AVGP	1.850	=	**1.833**	>	2.317	"
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	"

C_S, i.e. the description-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 9.2 is concerned with stratified results.

When stratifying the results by base learner (the second block in Table 9.2), we notice a striking difference in average ranks between J48 and the rest.

Table 9.3: Average ranks of the base learners, with critical difference CD.

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

Restricted to the J48 results, 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 axis-parallel fragments. The descriptions the EMM algorithm finds in the Local Pattern Mining phase are constructed by several conditions on single attributes. Hence the domain of each description has a shape similar to a J48 decision boundary, unlike a (non-degenerate) SVM decision boundary. Hence, the expected performance gain when adding such descriptions 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^{rbf} method, though the difference with the combined attributes is not statistically significant. The remaining base learners benefit from the added descriptions. Notably, when using LibSVM^{rbf}, it is possible to rely only on the description-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 three datasets *Emotions*, *Scene*, and *Yeast*, evaluated with the measures introduced in Section 9.4. The average ranks of the base

learners over these 180 test cases can be found in Table 9.3; again, the Friedman test is easily passed. The Nemenyi test shows that J48 performs significantly worse than all SVM methods and that MetaLibSVM^{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 [36], which means that an advantage of the RBF-kernel over the linear kernel can be expected. Moreover, the non-extensively optimized LibSVM^{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 similarity in form of the descriptions and the J48 decision boundary particularly damages 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 9.2. 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^{rbf} by itself this is not the case.

When stratifying the results by quality measure (the third block in Table 9.2) 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 9.2 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 9.2, 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 peak at O∪S∪C (rank 2.84) followed by O∪S (3.34), O∪C (3.53), O (3.6), S (3.89) and S∪C (3.97) (significant difference only between the first and each of the last combinations). Hence, adding C has an effect on performance similar to the effect of adding S, and BR particularly benefits if both are added, demonstrating that the locally exceptional descriptions provide additional information on the label dependencies, not covered by C.

In the last block of Table 9.2 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.

9.5.3 Evaluation of the Decompositive Approaches

We can learn more from Table 9.2 when more blocks are additionally stratified by evaluation measure. Indeed, the decision of the attribute base does not seem to have an 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 9.4 demonstrates, the situation varies 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 stands to reason, since both approaches are dedicated to the prediction of a correct labelset. In fact, LP only predicts label sets previously seen in the training data. CC behaves similarly: if we consider only the additional attributes from the previous predictions (i.e. attribute set C), then we find that CC behaves similar to a sequence tagger. That is to say, 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^{rbf}.

Table 9.4: 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^{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**	0.700
Rec	2.156	=	**1.956**	=	2.422	>	3.467	0.700
AvgP	**1.000**	>	2.778	=	3.111	=	3.111	0.700
Rank	**1.000**	>	2.622	=	3.133	=	3.244	0.700
all & MetaLibSVM^{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**	1.563
Rec	2.778	<	**1.000**	=	2.333	=	3.889	1.563
AvgP	**1.000**	=	2.111	=	3.444	=	3.444	1.563
Rank	**1.000**	=	2.000	=	3.333	=	3.667	1.563

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^{rbf}.

Contrasting behavior to BR is shown by LP, which dominates recall, especially for MetaLibSVM^{rbf}, but completely neglects precision. This indicates a preference for predicting the more rare large label sets. The best balance between precision and recall is shown by CLR, even for MetaLibSVM^{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 for ranking loss, which are clearly dominated by CLR's ability to produce a high density of relevant labels at the top of the the rankings. The high recall of LP corresponds to good ranking losses, but the low ranks of BR show that its high precision is not due to a good ranking ability. This behavior was already observed, e.g. in [36] and [74], where BR often correctly pushed a relevant class to 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 equal weight to the five selected measures, we observe that CLR significantly outperforms the second-placed LP if all base learners are considered, and slightly loses against LP if MetaLibSVM^{rbf} is used (top row in both blocks in Table 9.4).

9.5.4 Efficiency

Apart from the unsatisfactory performance of J48 compared to SVM approaches, Table 9.3 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 description-based 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^{rbf}, the difference was more pronounced, with a reduction of 60%, 52% and 50%, respectively. We obtained a similar picture for LibSVM^{lin} on two datasets, with a reduction of even 82% (*Emotions*) and 65% (*Scene*). On the *Yeast* dataset, however, training C_S surprisingly takes almost 7 times longer than training C_O. This case is very likely an exception since comparing LibSVM^{lin} and hence using different parameters shows again a clear reduction. The numbers for MetaLibSVM^{lin} and MetaLibSVM^{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 descriptions is relatively small. Also, note 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.

9.6 Discussion and Related Work

As we discussed in Section 3.3, exhaustive Local Pattern Mining methods exist. In this chapter, we have selected the Exceptional Model Mining instance with the structure of a Bayesian network model as target concept, to fulfill the Local Pattern Mining phase in the LeGo framework. For this method, no exhaustive approach exists. We expect little disadvantage from using heuristic rather than exhaustive search in the Local Pattern Mining phase. The found descriptions 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 Mining phase.

The applied Local Pattern Mining algorithm was created to find descriptions 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 attempt at testing the utility for classification of the result of such a stand-alone multi-label description discovery process. Some recent sophisticated classifiers, for instance the multi-label lazy associative classifiers [110], are also based on local patterns. However, these patterns serve only the classifier: interpretation is not considered. Hence the different phases, as present in the LeGo framework, are not as separated as they are in our work. Similarly, Cheng and Hüllermeier [11] 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 [44] use the outputs of a BR-SVM classifier as additional input attributes for second-level SVMs. Similarly, Tsoumakas et al. [104] 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 [92] rely on stacking the outcomes of the predetermined sequence of previous binary relevance classifiers, which permits modeling conditional dependencies, but it does not rely on locality. Zhang and Zhang [118] 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 parents as additional attributes. The very recent LIFT algorithm selects particularly representative centroids in the positive and negative records of a class by k-means clustering and then replaces the original attributes of a record by the distances to these representatives [117]. 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.

9.7 Conclusions

We have proposed enhancing multi-label classification methods with local patterns in a LeGo setting. These descriptions 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 targets. Hence each delivered description represents a local anomaly in conditional dependence relations between targets. Each description corresponds to a binary attribute which we add to the dataset, to tentatively 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 descriptions 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 descriptions found by a search process that is not at all concerned with the potential of the descriptions for classification, but is focused on exceptionality. In fact, the description 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 attempt at discovering multi-label descriptions 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 φ_{ed} from Section 6.3, or other search strategies. In partic-

ular, optimizing the beam search in order to properly balance its levels of exploration and exploitation, could fruitfully produce a more diverse set of attributes [70] in the Local Pattern Mining phase. Alternatively, description diversity could be addressed in the Pattern Subset Discovery phase, ensuring diversity within the subset S rather than enforcing diversity over the whole description 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 [97]. Also, experimentation on more datasets seems prudent. In this chapter, we have experimented on merely three datasets, selected for having a relatively low number of labels. As stated in Chapter 6, 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 decided 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 chapter should not be taken as irrefutable proof, but more as evidence contributing to our beliefs.

Acknowledgments

The research in this chapter is financially supported by the German Science Foundation (DFG). We also gratefully acknowledge support by the Frankfurt Center for Scientific Computing.