Propositionalization and Redundancy Treatment

Mark-A. Krogel and Stefan Wrobel

Otto-von-Guericke-Universität, Magdeburg, Germany
{krogel,wrobel}@iws.cs.uni-magdeburg.de

Abstract. Following the success of relational learning/inductive logic programming on structurally complex but small problems, recently there has been strong interest in relational methods that scale to real-world databases (relational data mining). Transformation-based methods have already been shown to be particularly promising approaches for robustly and effectively handling larger relational data sets. However, these methods also pose problems: beside the cost of joins, they can produce very large numbers of features (attributes), and among those there are possibly higher proportions of redundant features. In this paper, we investigate the extent of redundancies and approaches to dealing with them. We present promising results from experiments in several domains.

1 Introduction

Relational databases are a widespread and commonly used technology for storing information in business, administration, and science, among other areas. They pose special challenges to knowledge discovery because many prominent learning systems can only treat attribute-value or propositional data and leave transformations of relational data into that form to experienced analysts. In contrast, methods from inductive logic programming (ILP) can directly handle problems of relational data analysis. In particular, transformation-based approaches, which automatically transform relational data into a form accessible to propositional learners, have shown to be a robust and powerful method for handling relational databases (cf. results of KDD Cup 2001 [2]). The transformation process is also often refered to as propositionalization.

Our system RELAGGS (RELational AGGregationS) is a variant of such a transformation-based approach that relies on facilities inspired by SQL aggregation functions [7]. The approach encompasses facilities to avoid redundancies during early phases of propositionalization. Even so, we were inspired by irrelevant feature elimination as presented by [10] and by another approach to propositionalization with the help of aggregation [5] to further investigate issues of redundancy here. We demonstrate the results of this investigation with the help of 9 variants of learning tasks from 6 different domains, which are well-known in the fields of machine learning and knowledge discovery in databases. In effect, a learner could be derived that is more efficient without sacrificing accuracy.

The paper is organized as follows. Section 2 provides an introduction to our approach to propositionalization and view of redundancy issues. In Section 3, we give a short overview of redundancy treatment as presented by other authors and our ideas for their enhancement. We present a detailed experimental evaluation of the different approaches to redundancy treatment in Section 4. Section 5 concludes and gives pointers to future work.

2 Redundancy in the Context of Propositionalization

We gave a formal description of a framework for transformation-based learning in [7]. This framework may accommodate approaches as manifold as the pioneering ILP approaches to transformation-based learning LINUS and DINUS [9], extensions thereof [10], and also statistical approaches [6], among others. An extension of this framework led to the development of the system RELAGGS. The approach is described in this section from a database perspective since we believe that this could be an interesting viewpoint as a completion to the presentation from the ILP perspective as provided in [7]. Furthermore, usage of database management systems (DBMS) seems a valuable endeavor, especially in order to deal with larger data sets.

Propositionalization is intended to help in solving common learning tasks concerning relational data sets. Usually, such a learning task specifies one of the tables of the database as the target table, which contains an attribute of interest, the target attribute. A model has to be learned that determines the values of the target attribute based on the values for the other attributes of the target table and the other tables. In this paper, we deal with binary target attributes. This can be easily extended to multi-class problems and also to regression problems.

RELAGGS takes as input a database D containing a number of relations or tables and their descriptions (schemas) including information about attribute names and types. In addition, one of the attributes has to be marked as the target attribute, which also qualifies the corresponding table as the target table. The target table is taken to describe one example or instance per line. From foreign key information, the user can construct an ordered set of foreign links L that specify possible natural joins originating in the target table. Foreign links were introduced by [15] and further investigated in [16].

L induces a tree on the tables of D with the target relation as its root. Each subtree with the same root specifies a possibility to join the corresponding tables. With the parameter branching factor set to 0, not all these subtrees are considered, but only paths within these subtrees having the target relation as their first node. RELAGGS performs these joins for each instance given in the target table. The join results are then treated corresponding to the types of attributes contained. Equivalent functions to SQL avg, max, min, and sum are applied to numeric columns, while for nominal values, a new column is introduced for each possible value, and these columns are filled with the number of occurrences of the possible values. If the parameter cardinality is set to n, nominal attributes with more than n distinct possible values are ignored. In addition, the sizes of

the joins are recorded. Thus, for each join and instance, a single tuple of values is produced. These are concatenated in the same order as the elements of L into one tuple for each instance and combined with the identifier (for reasons of reference) and with the class information of the instance. The set of these tuples form the output of RELAGGS.

Example 1. As an example, we take the East-West challenge [11, 10]. In this domain, we can easily demonstrate the kind of representation of the data chosen as the input for RELAGGS and the effects of the bias that results from using aggregation functions. This way, comparisons to other approaches may be simplified. The learning task is to discover models that classify trains as eastbound or westbound. The problem is illustrated in Figure 1.

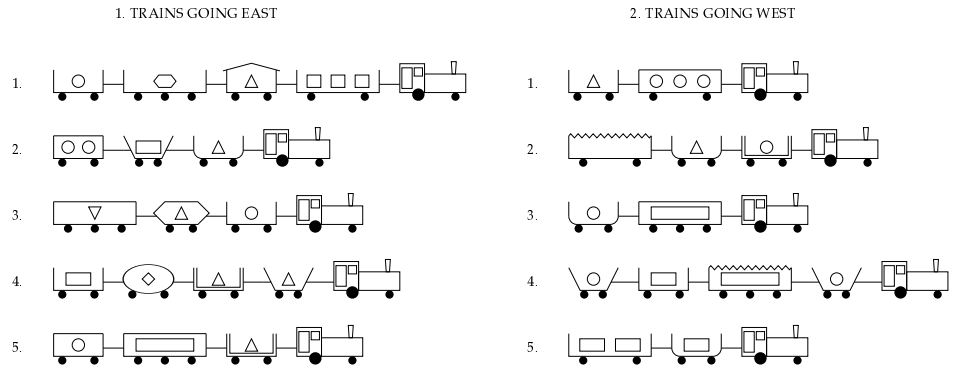

Fig. 1. The ten train East-West challenge [11, 10]

We chose to represent the information about trains in a relational database that contains 3 relations (tables) as depicted in Figure 2. Tables are shown here with their name in the first line, the attribute names in the second, and the values of the attributes below. Dashed arrows indicate foreign key relationships, always pointing from the foreign key attribute to the primary key attribute. The target attribute is **bound**.

From the foreign key relationships, two foreign links can be derived, as depicted by the solid arrows in Figure 2. The tree induced by the foreign links gives rise to natural joins with the following tables in the from-part of the corresponding SQL clauses:

J1: train, car
J2: train, car, load

These joins are computed first for the train with t_id = 1. The numeric and categorical attributes of those joins are aggregated and thus transformed into

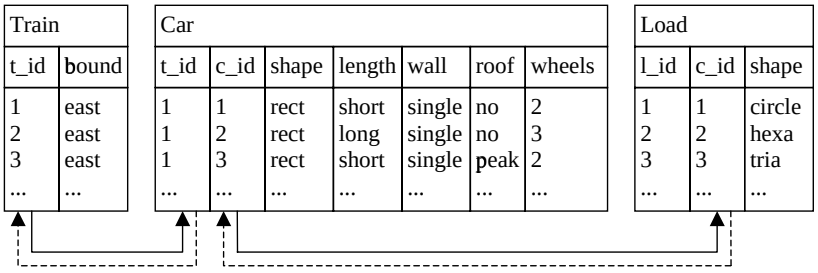

Fig. 2. A relational database for trains

single lines, which are in turn concatenated and enriched with identifier and class information for the example at hand. The tuples for all the examples are finally the RELAGGS output as shown in Table 1.

Table 1. The propositionalized trains

t_id	count_J1	car_shape_rect_J1	...	car_wheels_sum_J1	...	load_shape_circle_J2	bound
1	4	4	...	9	...	1	east
2	3	1	...	6	...	2	east
...	...	...	...	...	...	...	...
10	2	1	...	4	...	0	west

When we use the concept redundancy here, we do not refer to duplication of information as it is in many cases the motivation for normalization of relational databases. In a stricter sense, we understand redundancy here first of all as identical columns in the propositionalized table. In a broader sense, we also include into this concept of redundancy certain irrelevant parts of the data and knowledge representations, which are not able to contribute to the models to be learned or to their performance.

Those redundancies may occur at different stages of propositionalization processes. They may concern the original relational data, their propositionalization, and also the models derived from propositionalized data. The main focus and starting point here are redundancies within propositionalized tables. However, the aim should be to detect redundancies as early as possible to arrive at more efficient approaches to propositionalization.

Beside identical columns in the propositionalized tables that can be regarded as redundant, as mentioned above, there are also the trivial cases of columns containing just one distinct value, possibly mixed with NULL values, such that these attributes cannot contribute to the effective discrimination of positive and negative examples. Furthermore, attributes that are not completely identical but highly correlated may also be taken as redundant.

3 Approaches to the Treatment of Redundancy

3.1 An Approach in the LINUS Tradition

Within our approach to propositionalization, we had already proposed to make use of functional dependencies between relations in order to avoid redundancies that occur when applying local propositionalization functions [7]. This should be enhanced by further measures to avoid redundancy as will be demonstrated in the experimental section below.

One line of research into dealing with irrelevancy can be found in the context of work on the transformation-based learner LINUS [9] and extensions thereof [10]. While in LINUS, the focus is on the elimination of irrelevant literals and clauses within the models learned, the extended transformation approach mentioned above concentrates on the treatment of propositionalized tables. The approach is restricted to boolean tables, i.e. with truth-values for the attributes. Attributes can be identified as irrelevant based on their coverage of pairs of positive and negative examples.

It seems worthwhile to extend at least parts of that approach to the more general case of numerical instead of boolean tables. However, the exhaustive treatment of pairs of positive and negative examples will pose problems for larger data sets. Here, we do not adopt this strategy. Instead, as a first step to find out about the extent of redundancy, we use an algorithm that we proposed earlier for unsupervised feature selection and that we call FEASEBLE (FEAture SElection By cLustEring) [8], cf. Table 2.

Table 2. FEASEBLE algorithm

1. **Accept** as input: [-1, 1]-range-normalized propositionalization result R_{in}, number of steps s, allowed deviation d
2. **Initialize** a list L_1 of similar attributes with all attribute names from R_{in}, initialize L as list of lists of similar attributes with the single element L_1, initialize $L_{newlist}$ with empty list
3. **For** i := 1 to s
   (a) **Draw** an instance I_i from R_{in} at random without replacement
   (b) **For** all elements L_k of L
      i. **For** all elements of L_k
         A. **Compute** $L_{k\_newlist}$ as list of sublists from L_k where all elements of each sublist, i.e. corresponding attribute values for I_i, fulfill condition of difference less or equal to d
         B. **Add** elements of $L_{k\_newlist}$ to $L_{newlist}$
   (c) **Replace** L by $L_{newlist}$, empty $L_{newlist}$
4. **Output** first element of each list in L

This algorithm originated from attempts to detect identical columns in propositionalized tables, i.e. strictly redundant features. A first naive approach of a

pairwise comparison of columns of tables handled by DBMS turned out to be not efficient enough with the program running for several hours on the propositionalized PKDD99-00 financial data set that is described below. With the new algorithm, the detection of redundant attributes could be accomplished in a few minutes. Parameter settings have to be chosen here as s = number of examples and $d = 0$. We call this application FEASEBLE-red here in order to distinguish it from former different applications.

Note that although FEASEBLE-red first of all eliminates copies of identical columns from propositionalized tables, it also alleviates situations with one-value columns and highly correlated columns because of range-normalization accomplished beforehand by RELAGGS [7]. Results of the application of FEASEBLE-red are intended to lead to information about the sources of redundancies within the original data sets.

3.2 An Approach in the Line of Polka

For the system Polka [5], the same approach of using aggregate functions during propositionalization was taken as for RELAGGS. However, there are differences in the algorithmic approaches. RELAGGS first computes for each example all (natural inner) joins that follow from the foreign link declarations. Then, these joins are compressed with the help of aggregate functions. The results of compression are concatenated as sketched above.

Polka also uses a graph structure induced onto the set of relations by their foreign key relationships. A parameter d is used to bound the length of paths along which relations are used. This is here especially important in case of cycles in the graph. Then, the relations on these paths that are most distant from the target relation are first aggregated and then joined to their parents with the help of outer joins, in a recursive way, such that the result is again a propositionalized table. This procedure is more efficient than RELAGGS wrt. both time and space, however, Polka may miss relevant information since it does not use all possible joins.

Concerning this last point, the question arises if this is really an important issue for empirical data analyses. Below, we show that the effects of using less information as done by Polka are not strong and even positive. That is why we propose an even more radical approach, viz. the usage of each relation just once (ERJO) in the sense that the attributes of relations occuring in several joins as used by RELAGGS so far are only used by aggregate functions at the occasion of their first occurrence. Following the example presented above, the car relation would be used for propositionalization just once, viz. based upon its first occurence, which is in join J1.

Behind this procedure, there are the assumption that each relation has the strongest influence on the examples when not weighted by relations following it in the induced graph, and the assumption that those weighted attributes would be redundant in the weak sense that they often in practice can not contribute significantly to the models to be learned. Table 3 shows a sketch of the proposed algorithm

Table 3. A sketch of the RELAGGS-ERJO algorithm

1. **Accept** as input: database D, declarations of target attribute t and foreign links L
2. **Initialize** two sets J_{out} and J with the target relation schema containing t from D, and a set R with all other relation schemas from D, and let $J_{before} = \emptyset$ and $R_{handled} = \emptyset$
3. **While** $R \neq \emptyset$ and $J \neq J_{before}$ do
   (a) **Let** $J_{before} = J$
   (b) **Determine** all relations in R reachable via foreign links from L in one step from the last relations of join definitions in J and add their schemas to $R_{handled}$
   (c) **Add** elements of $R_{handled}$ once as last elements to appropriate joins in J
   (d) **Add** elements of J to J_{out}
   (e) **Remove** elements of $R_{handled}$ from R
   (f) **Let** $R_{handled} = \emptyset$
4. **Convert** elements of J_{out} to the form of clauses C with a literal produced from the target relation schema as head and the other relations of joins as literals in the body in the same order as in the join definitions
5. **Use** the resulting set of clauses $\mathcal{C}$ as done in the original RELAGGS algorithm [7], however, make use of only the last literal of each clause for propositionalization

We can do here without a depth parameter d since circles in the graph pose no problems and the number of relations of D is finite. This is also an advantage over the original RELAGGS that could only treat trees induced on the set of relations by foreign link declarations. However, a depth parameter d might be useful based on the assumption that relations closer to the target relation should have a higher influence and more distant relations could be included into the analysis on a step by step basis with further advantages for the learner's performance.

4 Experimental Evaluation

We evaluated the effects of redundancy treatment for propositionalization with the help of 9 data sets/learning tasks from 6 domains. All the learning tasks aim at predictive models for binary classification.

4.1 Data sets and Learning Tasks

The Trains domain and learning task is described above, cf. Example 1.

For the Mutagenesis problem, which evolved into one of the most important ILP benchmarks, [13] present among others a variant of the original data named NS+S2 that contains information about chemical concepts relevant to a special kind of drugs, the drugs' atoms and the bonds between those atoms. This variant is also known in the literature as B4. We could establish a single relation describing drugs. This single table joins and thus replaces a number of two-argument predicates with exclusively one-to-one relationships of the items

described. Finally, three relations (drugs, atoms, bonds) are the input for RELAGGS. The Mutagenesis learning task is to predict whether a drug is mutagenic or not. The separation of data into "regression-friendly" (188 instances) and "regression-unfriendly" (42 instances) subsets as described by [13] is kept here.

For the KDD-Sisyphus I Workshop at the ECML98, a data set based on a data warehouse of a Swiss insurance company was issued, described and investigated on several occasions such as [4]. The data within 10 relations describe the company's customers ("partners"), their households and their insurance contracts. Two learning tasks were provided with the data; they involve learning classifications of partners and households. For RELAGGS with branching factor 0 as used in [7], 12,772 partners (those of class 1 or 2 and with household information) and 7,329 households (those of class 1 or 2) were treated. We had to apply a further parameter here, viz. cardinality of categorical attributes, set to 50, in order to arrive at column numbers that could be handled by the DBMS used in the process of analysis and its preparations. Further, we use random samples from the partner and household tables, representing 997 instances each.

The PKDD Challenges in 1999 and 2000 offered a data set from a Czech bank [1]. The data set comprises of 8 relations that describe accounts, their transactions, orders, and loans, as well as customers including personal, credit card ownership, and socio-demographic data. A learning task was not explicitly given for the challenges. We compare problematic to non-problematic loans regardless if the loan projects are finished or not. The data describes 682 loans.

The PKDD Challenges from 1999 to 2001 provided another data set originating from a Japanese hospital specialized in the treatment of collagen diseases. A description of the data set may be found in [17]. We concentrate on the group of patients followed by the hospital and provided with both laboratory and special examination results. This group includes 417 patients. RELAGGS was applied here with branching factor 0.

The KDD Cup 2001 [2] tasks 2 and 3 asked for the prediction of gene function and gene localization, respectively. From these non-binary classification tasks, we extracted two binary tasks, viz. the prediction whether a gene codes for a protein that serves cell growth, cell division and DNA synthesis or not and the prediction whether the protein produced by the gene described would be allocated in the nucleus or not. RELAGGS treated all 862 examples from the Cup's training set with branching factor 0.

4.2 Procedure

The experiments were carried out on a PC with a Pentium III/500 MHz processor and 128 MB RAM. We decided to utilize the data mining system WEKA [14] with data prepared by our Java tools that in turn process data stored by the DBMS MySQL. Both WEKA and MySQL are freely available on the Web. The SQL scripts and ARFF files as well as Java code relevant for the presented experiments are available on request from the first author.

As input to the experiments, we take the results produced by RELAGGS. We include the application of WEKA's ZeroR to the input data. This provides a baseline for comparisons equivalent to the default error rates that occur when the more frequent class is always assumed as well as a basis for t-tests.

The main learning algorithm from WEKA that we apply in the experiments is J48, which largely corresponds to C4.5 [12]. J48 is applied directly to the RELAGGS results ("J48direct"). This is compared to the application of J48 to the RELAGGS results that were treated before by FEASEBLE-red and ERJO, respectively. All WEKA tools are applied with default parameter settings, including stratified 10-fold cross-validation for classification learning. Exceptions are the treatment of the domains Trains and Mutagenesis where we applied a leave-one-out scheme for learning with WEKA's J48 and PART, a rule induction algorithm.

4.3 Results

For Trains/Train.bound, PART produced the following rule from the RELAGGS result: if load_shape_tria_J3 > 0 and car_wall_single_J2 > 2 then east else west. This rule misclassifies train 5. FEASEBLE-red (redundancy elimination) and ERJO (each relation just once) did not produce different results for Trains.

For Mutagenesis/Drug42.active, the J48 model reached a prediction accuracy of 83.3%, while ERJO achieved 85.7%. For Mutagenesis/Drug188.active, the J48 model arrived at a prediction accuracy of 85.1% and with ERJO at 90.4%. It had taken RELAGGS 28 seconds on a Sun UltraSPARC-II to arrive at the propositionalized table, and a single training run took less than 1 second, back on the PC.

Table 4 shows, for the remaining data sets and tasks and for each of the experimental conditions, the average error across the ten folds and the standard deviation; the best result on each task is marked in **bold**.

As a rule, RELAGGS conditions are significantly better than Zero-R according to a paired t-test at level $\alpha = 0.05$, but there are no significances of differences between RELAGGS conditions. There are a few exceptions to this rule, where there are no statistical significances to be observed at all such as for the PKDD99-01 Patient.thrombosis task. Differences of accuracy between the J48direct and FEASEBLE-red conditions are due to the special handling of NULL values that are taken here not to prevent equality of attribute values while there is at the same time a random component in the choice of attributes by FEASEBLE.

Table 5 shows the attribute numbers under the different conditions; the smallest number on each task is marked in **bold**.

Learning was faster under the ERJO condition compared to the J48direct condition. We did not yet use an implementation of the algorithm proposed above but eliminated the appropriate attributes using WEKA's AttributeFilter on the complete RELAGGS result. Further improvements of efficiency can be expected from the application of RELAGGS-ERJO. FEASEBLE-red was slower than the other approaches due to the time it took to eliminate redundant attributes here.

Table 4. Error rate averages and standard deviations from 10-fold cross-validation

Data set/Task	**ZeroR**	**J48direct**	**FEASEBLE-red**	**ERJO**
ECML98/	17.2	15.6	**15.0**	15.4
Partner.class	± 0.4	± 3.2	± 2.6	± 2.7
ECML98/	47.6	9.1	9.1	**9.0**
Household.class	± 0.4	± 2.9	± 2.7	± 2.8
PKDD99-00/	11.1	7.8	7.6	**6.2**
Loan.status	± 0.7	± 3.5	± 3.4	± 2.4
PKDD99-01/	15.8	**13.4**	**13.4**	**13.4**
Patient.thrombosis	± 1.1	± 4.7	± 4.7	± 4.7
KDD01/	31.9	**16.9**	17.2	**16.9**
Gene.fctCellGrowth	± 0.6	± 3.3	± 3.1	± 3.3
KDD01/	42.6	12.9	**12.8**	13.0
Gene.locNucleus	± 0.5	± 2.3	± 2.2	± 2.3

Table 5. Numbers of attributes

Data set/Task	**J48direct**	**FEASEBLE-red**	**ERJO**
Trains/Train.bound	45	33	**28**
Mutagenesis/Drug188.active	130	96	**73**
Mutagenesis/Drug42.active	130	81	**73**
ECML98/Partner.class	824	**517**	557
ECML98/Household.class	933	562	**531**
PKDD99-00/Loan.status	937	563	**159**
PKDD99-01/Patient.thrombosis	297	**263**	297
KDD01/Gene.fctCellGrowth	895	**586**	860
KDD01/Gene.locNucleus	895	**548**	860

We could observe a tendency to simpler models under both the FEASEBLE-red and ERJO conditions.

4.4 Discussion

For the Trains domain, the bias used for the LINUS extension [10] was obviously more effective yielding the better Prolog rule

```
eastbound(T,true):-hasShortCar(T),hasClosedCar(T).
```

Actually, without taking load into account, we arrived with PART at the following rule: if count_j1 $>$ 2 and car_roof_jagged $<=$ 0 then east else west. This rule makes no mistakes on the 10 trains and it demonstrates the value of counting, here for the number of cars per train, which was not included in the LINUS extension examples, and the way to express negation under the RELAGGS bias, here as a part of the rule stating that the train has no car with a jagged roof.

For the Mutagenesis tasks, our approaches reach accuracies in the same order as extended LINUS (83% on Drug42.active [10]) and Polka (up to 89% on Drug188.active [5], as well as other learners. To our knowledge, the RELAGGS approach is the most efficient so far [3].

In general, there was no loss in accuracy under the ERJO condition as it might have been expected, on the contrary. We understand this as an empirical justification of this approach as well as similar variants of propositionalization as instantiated by Polka.

The missing efficiency of FEASEBLE-red in combination with the partly significant numbers of redundant attributes in the propositionalized tables points to the importance of the original objective of the investigation, i.e. the treatment of redundancy as early as possible. Apart from trivial cases such as one-value columns in the RELAGGS input data or not occurring categorical values despite of their enumeration in database documentations or because of sampling, redundancies in the RELAGGS output can be caused by more complicated situations such as one-to-one relationships of tables as manifested by the data contained in them, not by their foreign key relationships, or by certain categorical values occuring in the same extent for all examples, positive and negative. Here, further work has to be done to find out about efficient possibilities to treat these situations.

5 Conclusion

We have shown that redundancy is an important matter during propositionalization and pointed to approaches to its treatment. Further, we empirically justified heuristic approaches of non-exhaustive feature generation. This way, we could arrive at an improvement wrt. both accuracy and efficiency of our award-winning system RELAGGS.

In our future work, we plan to investigate a unified approach including both feature selection methods that we investigated elsewhere [8] and the methods

to deal with redundancy proposed in this paper. Furthermore, we will apply the extended LINUS [10] to data sets listed in Table 4 in order to allow for a more detailed comparison and possibly research into a combination of the types of bias used there and within our approach. We also intend to treat larger data sets bearing real-life properties with propositionalization, to include other aggregation functions into RELAGGS [7], and to apply sampling techniques during propositionalization to further increase scalability and as a step towards a variant of active learning.

Acknowledgements

We would like to thank the WEKA development team for their tool, which proved very useful and pleasant to work with for us. Special thanks to Nada Lavrač and Peter Flach for their support. Thanks to our colleague Susanne Hoche for a preparation of the Mutagenesis data for MIDOS and her C^2RIB, which could be reused for the application of RELAGGS. This work was partially supported by the DFG (German Science Foundation), project FOR345/1-1TP6.

References

1. P. Berka. Guide to the Financial Data Set. In A. Siebes and P. Berka, editors, *PKDD2000 Discovery Challenge*, 2000.
2. J. Cheng, C. Hatzis, H. Hayashi, M.-A. Krogel, S. Morishita, D. Page, and J. Sese. KDD Cup 2001 Report. *SIGKDD Explorations*, 3(2):47–64, 2002.
3. S. Hoche and S. Wrobel. Scaling Boosting by Margin-Based Inclusion of Features and Relations. In *submitted*, 2002.
4. J.-U. Kietz, R. Zücker, and A. Vaduva. MINING MART: Combining Case-Based Reasoning and Multistrategy Learning into a Framework for Reusing KDD-Applications. In R. S. Michalski and P. Brazdil, editors, *Proceedings of the Fifth International Workshop on Multistrategy Learning (MSL)*, 2000.
5. A. J. Knobbe, M. de Haas, and A. Siebes. Propositionalisation and Aggregates. In L. de Raedt and A. Siebes, editors, *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Disovery (PKDD)*. Springer-Verlag, 2001.
6. S. Kramer, B. Pfahringer, and C. Helma. Stochastic Propositionalization of Non-Determinate Background Knowledge. In D. Page, editor, *Proceedings of the Eighth International Conference on Inductive Logic Programming (ILP)*. Springer-Verlag, 1998.
7. M.-A. Krogel and S. Wrobel. Transformation-Based Learning Using Multirelational Aggregation. In C. Rouveirol and M. Sebag, editors, *Proceedings of the Eleventh International Conference on Inductive Logic Programming (ILP)*. Springer-Verlag, 2001.
8. M.-A. Krogel and S. Wrobel. Propositionalization and Feature Selection. In *submitted*, 2002.
9. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1993.

10. N. Lavrač and P. A. Flach. An extended transformation approach to Inductive Logic Programming. *ACM Transactions on Computational Logic*, 2(4):458–494, 2001.
11. R. S. Michalski. Pattern Recognition as Rule-guided Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):349–361, 1980.
12. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
13. A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: a study in first-order and feature-based induction. *Artificial Intelligence*, 85(1,2):277–299, 1996.
14. I. H. Witten and E. Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.
15. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In J. Komorowski and J. Żytkow, editors, *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD)*. Springer-Verlag, 1997.
16. S. Wrobel. Inductive Logic Progamming for Knowledge Discovery in Databases. In N. Lavrač and S. Džeroski, editors, *Relational Data Mining*. Springer-Verlag, Berlin, New York, 2001.
17. J. Żytkow and S. Gupta. Guide to Medical Data on Collagen Disease and Thrombosis. In P. Berka, editor, *PKDD2001 Discovery Challenge on Thrombosis Data*, 2001.