Local Patterns: Theory and Practice of Constraint-Based Relational Subgroup Discovery

Nada Lavrač[1], Filip Železný[2], and Sašo Džeroski[1]

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
nada.lavrac@ijs.si
saso.dzeroski@ijs.si
[2] Czech Technical University, Prague, Czech Republic
University of Wisconsin, Madison, USA
zelezny@biostat.wisc.edu

Abstract. This paper investigates local patterns in the multi-relational constraint-based data mining framework. Given this framework, it contributes to the theory of local patterns by providing the definition of local patterns, and a set of objective and subjective measures for evaluating the quality of induced patterns. These notions are illustrated on a description task of subgroup discovery, taking a propositionalization approach to relational subgroup discovery (RSD), based on adapting rule learning and first-order feature construction, applicable in individual-centered domains. It focuses on the use of constraints in RSD, both in feature construction and rule learning. We apply the proposed RSD approach to the Mutagenesis benchmark known from relational learning and a real-life telecommunications dataset.

1 Introduction

Inductive databases [11] embody a database perspective on knowledge discovery, where knowledge discovery processes are considered as query processes. In addition to normal data, inductive databases contain patterns (either materialized or defined as views). Data mining operations looking for patterns are viewed as queries posed to the inductive database. In addition to patterns (which are of local nature), models (which are of global nature) can also be considered.

A general formulation of data mining [19] involves the specification of a language of patterns and a set of constraints that a pattern has to satisfy with respect to a given database. The constraints that a pattern has to satisfy can be divided in two parts: language constraints and evaluation constraints. The first only concern the pattern itself, the second concern the validity of the pattern with respect to a database.

1.1 Constraints in inductive databases

Inductive queries consist of constraints and the primitives of an inductive query language include language constraints (e.g., find association rules with item A in the head) and evaluation primitives. Evaluation primitives are functions that express the validity of a pattern on a given dataset. We can use these to form evaluation constraints (e.g., find all item sets with support above a threshold) or optimization constraints (e.g., find the 10 association rules with highest confidence).

Constraints thus play a central role in data mining and constraint-based data mining is now a recognized research topic [4]. The use of constraints enables more efficient induction as well as focussing the search for patterns on patterns likely to be of interest to the end user. While many different types of patterns have been considered in data mining, constraints have been mostly considered in mining frequent itemsets and association rules, as well as some related tasks, such as mining frequent episodes, Datalog queries, molecular fragments, etc. Few approaches exist that use constraints for other types of patterns/models, such as size and accuracy constraints in decision trees [10] or in classification rule discovery.

1.2 Constraints in relational subgroup discovery

In this paper, we consider the use of constraints in the context of relational subgroup discovery (RSD). We consider the task of subgroup discovery defined as follows: given a population of individuals and a specific property of those individuals that we are interested in, find population subgroups that are statistically 'most interesting', e.g., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest [32]. We restrict ourselves to class-labeled data in our approach, with the class attribute being the property of interest.

While the goal of standard rule learning is to generate models, one for each class, inducing class characteristics in terms of properties occurring in the descriptions of training examples, in contrast, subgroup discovery aims at discovering individual 'patterns' of interest. In this sense, subgroup discovery belongs to *descriptive induction* [23, 34] which has recently gained much attention of researchers developing rule learning algorithms. These involve mining of association rules (e.g., the APRIORI association rule learning algorithm [1]), clausal discovery (e.g., the CLAUDIEN system [23, 24]), subgroup discovery (e.g., the MIDOS [32, 33], EXPLORA [12], SD [9] and CN2-SD [17] subgroup discovery systems) and other approaches to non-classificatory induction aimed at finding interesting patterns in data.

Our approach to constraint-based RSD first performs feature generation, then applies a propositional approach to subgroup discovery (the RSD implementation in the Yap Prolog with a user's manual and sample problems can be obtained from `http://labe.felk.cvut. cz/~zelezny/rsd`). The combination of the above mentioned strategies controlled by constraints represents an original approach to relational subgroup discovery, altough previous work exists incorporating some of the techniques mainly in classification rule discovery; e.g., rule induction with constraints in relational domains including propositionalization [2, 3], or using rule sets to maximize ROC performance [7].

1.3 Outline of the paper

This paper investigates local patterns in the multi-relational constraint-based data mining framework. Given this framework, it contributes to the theory of local patterns by providing the definition of local patterns and proposing a set of objective and subjective measures for evaluating the quality of induced patterns (Section 2). These notions are applied to a description task of subgroup discovery, for which a practical relational subgroup discovery algorithm RSD has been developed (Section 3). Section 4 discusses the use of constraints in RSD, followed by the experimental evaluation of the proposed approach to subgroup discovery in Section 5.

2 Theory of local patterns

This section contributes to the theory of local patterns by providing the definition of local patterns, and proposing a set of objective and subjective measures for evaluating the quality of induced patterns.

2.1 Pattern discovery as rule learning

As in classification rule learning, we consider patterns of the form of a (backwards) implication:

$$Class \leftarrow Cond$$

Having limited the form of patterns to the above rule form, we limit the scope of investigation to patterns with a certain property of interest which is the goal of investigation (the target class, $Class$) that appears in the rule consequent. In the selected formalism the rule antecedent ($Cond$) is a conjunction of features (attribute-value pairs) selected from the features describing the training instances.

In the given scope, pattern discovery is a task at the intersection of predictive and descriptive induction. By inducing rules from labeled training instances (labeled positive if the property of interest holds, and negative otherwise), the process of subgroup discovery is targeted to uncovering properties of a selected target population of individuals with the given property of interest. In this sense, pattern discovery is a form of *supervised learning*. The fact that a pattern discovery task aims at characterizing population subgroups of a given target class suggests that standard classification rule learning could be used for solving the task. However, pattern discovery is a form of *descriptive induction* as the task is to uncover individual rules or patterns of interest, which must be represented in explicit symbolic form and which must be relatively simple in order to be recognized as actionable by potential users.

Each pattern can be extended with the information about the rule *quality*. Unlike in association rule learning, where rules are equipped with the *support* and *confidence* of a rule, in this paper a standard rule pattern has the following form:

$$Class \leftarrow Cond\ [TPr, FPr] \tag{1}$$

where $Class$ is the target property of interest, $Cond$ is a conjunction of features (attribute-values), TPr is the *true positive rate* or the *sensitivity*, computed as $p(Cond|Class) = \frac{n(Class.Cond)}{Pos}$, and FPr is the *false alarm* or *false positive rate*, computed as $p(Cond|\overline{Class}) = \frac{n(\overline{Class}.Cond)}{Neg}$. In these formulas $n(Class.Cond)$ is the number of true positives TP (the number of covered instances belonging to $Class$), $n(\overline{Class}.Cond)$ the number of false positives FP (the number of covered instances not belonging to $Class$), Pos is the number of positives (instances of the target class), Neg the number of negatives, and $N = Pos + Neg$ is the size of the entire population.

2.2 Pattern evaluation measures

One can distinguish between *objective* and *subjective* quality measures (measures of interestingness) [26]. Both the objective and subjective measures need to be considered in order to solve pattern discovery tasks. Which of the quality criteria are most appropriate depends on the application. Obviously, for automated rule induction it is only the objective quality criteria that apply. However, for evaluating the quality of induced patterns and their usefulness for decision support, the subjective criteria are more important, but also harder to evaluate.

As shown in Section 2.1, each rule can be extended with the information about the rule quality. While the basic information of rule quality is usually attached to the induced rule itself, as output of the learning algorithm, other quality measures are usually computed for a ruleset, in order to evaluate the output of the induction process as a whole, enabling the comparison of the performance of different algorithms.

Below is a list of *subjective* measures of interestingness:

- *Usefulness.* Usefulness is an aspect of rule interestingness which relates a finding to the goals of the user [12].
- *Operationability.* In this paper we have introduced the notion of operationability, which is one aspect of usefulness.
- *Actionability.* "A rule is interesting if the user can do something with it to his or her advantage" [25, 26]. Actionability is a special case of operationability.
- *Unexpectedness.* A rule is interesting if it is surprizing to the user [26].
- *Novelty.* A finding is interesting if it deviates from prior knowledge of the user [12].
- Redundancy. Redundancy amounts to the similarity of a finding with respect to other findings; it measures to what degree a finding follows from another one [12], or to what degree multiple findings support the same claims.

When discussing the *objective* quality measures - in line with the distinction between *predictive induction* and *descriptive induction* - one can distinguish between the *predictive* and *descriptive* quality measures. A typical predictive quality measure, measuring the quality of a ruleset, is *predictive accuracy* of a ruleset, defined as the percentage of correctly predicted instances.[1]

In contrast with predictive quality measures, descriptive quality measures evaluate each individual subgroup and are thus appropriate for evaluating the success of pattern discovery. The following measures turn out to be most appropriate for measuring the quality of individual rules: rule size, coverage, support, accuracy (in different contexts also called precision or confidence), significance and unusualness. Although the evaluation of each individual rule is of ultimate importance, their variants that compute the average over the induced set of subgroup descriptions enable the comparisons of subgroup discovery algorithms (see [17] for the exact definition of these measures).

To explain rule significance and unusualness, which are the most important pattern discovery measures, some of the other measures for evaluating the quality of rules of the form $Class \leftarrow Cond$ need to be explained first. *Coverage* $p(Cond)$ is a measure of *generality*, computed as the relative frequency of all the examples covered by the rule: $\frac{n(Cond)}{N}$. *Support* $p(Class.Cond)$ is computed as the relative frequency of correctly classified covered examples: $\frac{n(Class.Cond)}{N}$. Rule *accuracy* $p(Class|Cond)$ (called *precision* in information retrieval and *confidence* in association rule learning) is the fraction of predicted positives that are true positives. Next, we define *accuracy gain* as the difference between rule accuracy $p(Class|Cond)$ and default accuracy $p(Class)$ achieved by the trivial rule $Class \leftarrow true$.

- *Significance* of a rule is computed in terms of the likelihood ratio of a rule, normalized with the likelihood ratio of the significance threshold (99%). Significance (or *evidence*, in the terminology of [12]) indicates how significant is a finding if measured by this statistical criterion. In the CN2 algorithm [5], significance $Sig(Class \leftarrow Cond)$ is measured in terms of the likelihood ratio statistic[2] of a rule as follows:

$$2 \sum_i n(Class_i.Cond). \log \frac{n(Class_i.Cond)}{n(Class_i)} \qquad (2)$$

  where for each class $Class_i$, $n(Class_i.Cond)$ denotes the number of instances of $Class_i$ in the set where the rule body holds true, and $n(Class_i)$ is the expected number of $Class_i$ instances in a set chosen randomly from the entire instance set in $\sum_i n(Class_i.Cond)$ independent trials, so that

$$n(Class_i) = N_i \frac{\sum_i n(Class_i.Cond)}{\sum_i N_i}$$

  where N_i is the total number of $Class_i$ instances in the entire instance set. Note that although for each generated subgroup description one class is

[1] For a binary classification problem, ruleset accuracy is computed as $\frac{TP+TN}{N}$.

[2] In two-class problems this statistic is distributed approximately as χ^2 with one degree of freedom.

selected as the target class, the significance criterion measures the distributional unusualness unbiased to any particular class – as such, it measures the significance of rule condition only: $Sig(Class \leftarrow Cond) = Sig(Cond)$.

– *Unusualness* of a rule is computed by the *weighted relative accuracy* of a rule [15], defined as follows:

$$WRAcc(Class \leftarrow Cond) = p(Cond).[p(Class|Cond) - p(Class)]$$

Weighted relative accuracy can be understood as trading off rule *coverage* $p(Cond)$ and *accuracy gain* $p(Class|Cond) - p(Class)$.

As shown in [17], *WRAcc* is appropriate for measuring the unusualness of patterns, because it is proportional to the vertical distance from the diagonal in the ROC space (for ROC analysis, see [22]). As such, *WRAcc* also reflects rule significance - the larger *WRAcc* is, the more significant the rule is, and vice versa. As both *WRAcc* and rule significance measure the distributional unusualness of a pattern, they are the most important quality measures for pattern discovery, if the goal of pattern mining is— as is the case in this paper—finding of interesting population subgroups which are sufficiently large and distributionally unusual. However, while significance only measures distributional unusualness, computed in terms of correctly classified covered examples of all classes, *WRAcc* takes explicitly the rule coverage into the account, therefore we consider *unusualness* to be the most appropriate measure for pattern quality evaluation.

It can be shown that for a given pattern, its *WRAcc* value is proportional to the value of the Area Under the ROC Curve (AUC). Consequently, as optimizing *WRAcc* means also optimizing AUC, *WRAcc* proves to be of use not only as a heuristic appropriate for pattern discovery in descriptive induction, but also for predictive induction. This claim is supported by the results achieved in [29, 16] in the comparisons of variants of CN2 and CN2-SD in which *WRAcc* was used instead of the rule accuracy heuristic.

3 Background: Relational Subgroup Discovery

Our approach adapts classification rule learning to relational subgroup discovery, described in [16], achieved by (a) propositionalization through first-order feature construction, (b) incorporation of example weights into the covering algorithm, (c) incorporation of example weights into the weighted relative accuracy search heuristic, (d) probabilistic classification based on the class distribution of covered examples by individual rules, and (e) area under the ROC curve rule set evaluation. The main advantage of the proposed approach is that each induced rule with a high weighted relative accuracy represents a 'chunk' of knowledge about the problem, due to the appropriate tradeoff between accuracy and coverage, achieved through the use of the weighted relative accuracy heuristic.

The input to the RSD algorithm consists of *a relational database* containing one main table (relation), where each row corresponds to a unique *individual* and one attribute of the main table is specified as the *class* attribute - this table defines the *training examples*, and other tables (relations) defining the *background knowledge*. In addition, *a mode-language definition* is given, which is used to construct first-order features.

The output of RSD is a set of subgroups whose class distributions differ substantially from the class distribution in the complete data set. The subgroups are defined by conjunctions of (automatically generated/defined) first-order features. The RSD algorithm proceeds in two stages: first-order feature construction and rule-based subgroup discovery.

RSD First-order Feature Construction In our approach to first-order feature construction, based on [8, 13, 18], local variables referring to parts of individuals are introduced by so-called *structural predicates*. In a given language bias for first-order feature construction, a first-order feature is composed of one or more structural predicates introducing a new variable, and of *utility predicates* as in LINUS [14] (called *properties* in [8]) that 'consume' all new variables by assigning properties of individuals or their parts, represented by variables introduced so far. Utility predicates do not introduce new variables. (Examples of both types of predicates will be given below.)

The design of an algorithm for constructing first-order features can be split into two relatively independent problems:

Step 1: Identify features. This step results in identifying all first-order literal conjunctions that form a feature in the sense explained above, and at the same time comply to user-defined constraints (mode-language). Such features do not contain any constants and the task can be completed independently of the input data.

Step 2: Employ constants. This step results in extending the feature set by variable instantiations. Certain features are copied several times with some variables substituted to constants 'carefully' chosen from the input data. During this process, some irrelevant features are detected and removed, based on several constraints.

Both steps can be viewed as an exploitation of the combination of pre-set and user-defined sets of constraints of both syntactic (language-related) and semantic (data-oriented) character. From this viewpoint, they will be explained in detail in the devoted Section 4.

RSD Rule Induction Algorithm The core of RSD is a subgroup discovery algorithm which can accept data propositionalized by the feature constructor described above. The algorithm inherits some basic principles of the CN2 rule learner [5], which are adapted in several substantial ways to meet the needs of subgroup discovery. The principal improvements, making it appropriate for subgroup discovery, involve the implementation of the weighted covering algorithm, incorporation of example weights into the weighted relative accuracy heuristic, probabilistic classification, and the area under the ROC curve rule set evaluation [16].

4 Using constraints in RSD

The curse of combinatorial dimensionality is present in the principles underlying both procedural phases of RSD:

- We apply language constraints to define the language of possible subgroup descriptions. These are applied both in feature generation and rule induction.
- We apply evaluation constraints during rule induction to select the (most) interesting rules/subgroups.

Consequently, RSD makes heavy use of both syntactic and semantic constraints exploited by search-space pruning mechanisms. On one hand, some of the constraints (such as *feature undecomposability*) are deliberately enforced by the system and pruning based on these constraints is guaranteed not to cause the omission of any solution. On the other hand, additional contraints (e.g. maximum *variable depth*) may be tuned by the user. These are designed with the intention to most naturally reflect possible user's heuristic expectations or minimum requirements on quantitative evaluations of search results.

4.1 Constraints in feature construction

Motivated by language-bias declarations used in ILP systems, RSD accepts language declarations very similar to those used by the systems Aleph [27] and Progol [21], including variable types, modes, setting a *recall* parameter etc, used to syntactically constrain the set of possible features. The use of the language bias declarations are best explained on a simple example. For this purpose we use the well-known East-West trains domain [20].

Structural predicates. By the mode declaration `:-modeb(1, hasCar(+train, -car))` the user tells the system that the binary background relation `hasCar` may be employed in the body of constructed features, so as to provide the identification of some car of a specified train. The number 1 ("recall") determines that a feature can address at most one car of a given train. Input variables are labeled by the + sign, and output variables by the - sign.

Property predicates. Defined as above, but have no output variables.

Head predicate. Its declaration always contains exactly one variable of the input mode (e.g., `:-modeh(1, train(+train))`. The declaration serves merely to identify the key of the main individual.[3]

RSD produces all features satisfying the mode and setting declarations. The features produced by RSD have to satisfy an important constraint: a feature may not be decomposable into a conjunction of two features.

[3] The head declaration thus may seem overly complicated but contributes to compatability with declarations used with the widely used ILP systems mentioned earlier.

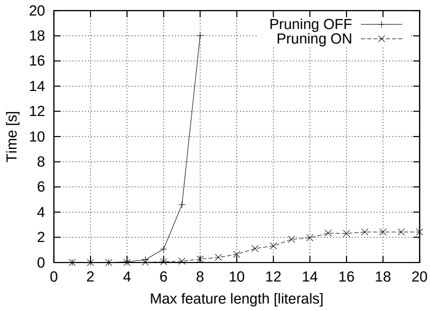

Fig. 1. The effect of pruning in the syntactic feature construction on efficiency in the East-West Trains domain. The diagram shows the amount of time needed to produce the exhaustive set of features for a given maximum feature length when pruning is off or on.

For example, the feature set based on the modes

```
:-modeh(1, train(+train)).
:-modeb(2, hasCar(+train, -car)).
:-modeb(1, long(+car)).
:-modeb(1, notSame(+car, +car)).
```

will contain a feature

```
f(A):- hasCar(A,B),hasCar(A,C),long(C),long(B),notSame(B,C).
```

but it will not contain a feature with the body

```
hasCar(A,B),hasCar(A,C),long(B),long(C)
```

as such an expression would clearly be decomposable into two separate features. We do not construct such decomposable expressions, as these are redundant for the purpose of the subsequent search for rules with conjunctive antecedents.

The language constraint of undecomposability plays a major role: it enables pruning the search for possible features without losing any solutions. As an example, Figure 1 illustrates the speedup gained by the pruning on the East-West Trains domain (evaluation on real-life data will be shown in the experimental section).

In addition, other language constraints can be specified. These are: the maximum length of a feature (number of contained literals), maximum *variable depth* [21] and maximum number of occurrences of a given predicate symbol. If constraints are not specified by the user, the first two acquire a default value while the last is unlimited.

Unlike Aleph and Progol declarations, RSD does not use the # sign to denote a constant-value argument. In the mentioned systems, constants are provided by a single saturated example, while RSD extracts constants from all the input data (examples). The user can instead utilize the special reserved property predicate `instantiate/1`, which does not occur in the background knowledge, to specify a variable that should be substituted with a constant during feature construction. For example, from the modes

```
:-modeh(1, train(+train)).
:-modeb(1, hasCar(+train, -car)).
:-modeb(1, hasLoad(+car, -load)).
:-modeb(1, hasShape(+load, -shape)).
:-modeb(*, instantiate(+shape)).
```

exactly one feature is generated:

```
f1(A) :- hasCar(A,B), hasLoad(B,C), hasShape(C,D), instantiate(D).
```

In the second step, after consulting the input data, `f1` will be substituted by a set of features, in each of which the `instantiate/1` literal is removed and the `D` variable is substituted with a constant making the body of `f1` provable in the data. Provided they contain a train with a rectangle load, the following feature will appear among those created out of `f1`:

```
f11(A) :- hasCar(A,B), hasLoad(B,C), hasShape(C,rectangle).
```

A similar principle applies for features with multiple occurences of the `instantiate/1` literal. Arguments of this literal within the feature form a set of variables ϑ; only those (complete) instantiations of ϑ making the feature's body provable on the input database will be considered.

However, not all such features will appear in the resulting set. For the sake of efficiency, we do not perform feature filtering by a separate postprocessing procedure, but rather discard certain features already during the feature construction process described above. The following constraints are used: (a) no feature should have the same value for all examples and (b) no two features should have the same values for all examples. For the latter case, only the syntactically shortest feature is chosen to represent the class of semantically equivalent features. In addition, a minimum number of examples for which a feature has to be true can be prescribed. This constraint is similar to the minimum support constraints in mining frequent item sets.

4.2 Constraints in subgroup discovery

In the subgroup discovery phase, a language constraint employed is the prescription of a maximal number of conditions/features in the description of a subgroup.

Several evaluation functions are considered. These include accuracy, weighted relative accuracy (WRAcc), significance, and area under the ROC curve. Accuracy and WRAcc are used in optimization constraints, i.e., RSD looks for rules with high accuracy/WRAcc. In fact, they are used as heuristic functions in RSD. Significance is used in evaluation constraints, i.e., one can prescribe a significance threshold that rules have to satisfy (expressed as significance at e.g., 99% level). WRAcc may be used in a similar fashion.

For lack of space we do not provide here a tabular summary of all employed constraints and the ways of their setting, this can be however found in the RSD user's manual available from the above mentioned RSD download page.

5 Experiments

We have experimented with the well-known relational learning benchmark concerning the Mutagenicity of chemicals and we have also applied RSD to the analysis of a real-life telecommunications dataset. The Mutagenesis data have been described in detail in many sources, see e.g. [28]. The Telecommunication application has been described by Železný et al. in [31, 30]; next we give a brief overview of the data.

5.1 Telecommunications

The data represent incoming calls (1995 items thereof) to an enterprize. Each such call is answered by a human operator and in the usual case further transferred to an attendant distinguished by his/her line number. Further re-transfers may also occur. Each sequence of such transfers is tracked by a computerized exchange and related data are stored in a logging file. By a suitable transformation thereof, one may obtain a relation `incoming/5`, represented by ground facts of the form `incoming(`*date, time, caller, operator, result*`).` The argument *result* either takes a constant value or is a recursively defined function, so that *result* $\in$ {`talk, unavailable, transfer([`$ln_1, ln_2, ..., ln_n$`],` *result*`)`}, where $ln_1 ... ln_{n-1}$ denote line numbers to which unsuccessful attempts to transfer have been made, and ln_n the result of the last transfer attempt.

For example, the ground fact

```
incoming(date(10,18), time(13,37,29), [0,6,4,8,2,5,6,8,4,9], 32,
transfer([16,12],transfer([26],talk))).
```

describes a call from the number 0648256849 at 13:37:29 on 10/18 received by the operator on line 32. The operator first tried to transfer the caller to line 16 without success, and then transferred him/her successfully to line 12. The person on line 12 further redirected the caller to line 26. After a talk with line 26, the call was terminated.

We divide all instances of incoming transferred calls into classes determined by the line to which the operator tried to transfer the caller first. We thus obtain 25 classes. Attributes of examples (the main table records) then consist of the first four arguments of `incoming/5` and the class attribute. Finding subgroups interesting with respect to this class attribute may contribute to purposes of decision support of the operator. Further, if the subgroup set has sufficient predictive power, it may partially or completely substitute the operator.

Let us now comment on two of the available background relations. The predicate `prefix(Number,Prefix)` is true whenever the second (output) argument is the prefix (of any length) of the first (input) argument. For instance, regarding the example given above, `prefix([0,6,4,8,2,5,6,8,4,9],[0,6,4])` is true.

This background predicate proved useful in previously published results, since it is able to bind callers from the same area, city, company, office etc. The predicate gives multiple possible outputs for a given input. When used as part of a feature definition, it will be the job of the feature constructor to decide which prefixes should be used (possibly in conjunction with other literals) to generate features with acceptable coverage measures. Out of the prefixes kept, the rule inducer chooses those that help identify interesting subgroups.

Another background predicate `prev_attempt/6` reflects the fact that a line desired by the caller may often be determined by looking at the caller's recent attempts to reach a person, i.e., by inspecting past records (w.r.t. the time-label of the current example) in the `incoming/4` relation. This problem setting is thus not far from what is known as *multi-instance learning* [6], where relevant attribute values describing an instance extend in multiple rows of a single table.

For example, the goal

```
prev_attempt(date(10,18),time(13,37,29),
[0,6,4,8,2,5,6,8,4,9], Line, When, Result).
```

will succeed with the result

```
Line=10, When=today, Result = unavailable,
```

provided the caller 0648256849 failed to reach line 10 on 10/18 before 13:37:29. Again, the `prev_attempt/6` may obviously yield multiple outputs for a given instantiations of the input arguments.

5.2 Expert analysis of induced subgroups: Evaluating novelty

We present the descriptions of some of the discovered subgroup in Telecommunication, with comments from the domain expert on the descriptions in Table 1 and the distributional characteristics of the subgroups.

Expert analysis of the induced rules shows that some of them identify novel and interesting information. Especially revealing are the comments related to the changes of class frequency associated with the rules. In the overall distribution, calls to line 21 are most common. The expert comments that this reflects his expectations, as the person at line 21 is a marketer, and people interested in products call this line most frequently. In subgroup `Tele1`, there is (a) an increase in line 21 frequency: clients not receiving an ordered package often wait until Friday and then complain with line 21; and (b) a decrease in line 13 frequency: the person at line 13 mostly collaborates with dealers who have less business on Fridays. For subgroup `Tele4` there is a) an increase in line 28 frequency: repeated attempts to reach line 28, and (b) an increase in line 21 frequency: the person at line 28 works as technical support for products sold by person on line 21.

The use of the undecomposability constraint and the pruning enabled thereby greatly reduces the time necessary to generate the features. This reduction increases with the maximum feature length, as illustrated in Figure 2.

5.3 Effects of constraints on feature generation

The use of the other feature constraints, i.e., the minimum coverage, unique coverage and incomplete coverage (the latter two are referred to as filtering) reduces the number of features generated, as shown in Figure 3. In Mutagenesis, the maximum feature length was set to 5 and the minimum feature coverage to 20 instances, obtaining 42 different features. In the Telecom domain, we set

Table 1. Subgroup descriptions in the form $H \leftarrow B\ [TP, FP]$, definitions of used features, and subgroup interpretation including expert's comments.

`Tele1`: `line21(A)` ← `f40(A)` [56,268] `f40(A):-call_date(A,B),dow(B,fri).` Calls received on Fridays. *Expert's evaluation*: Not a novel information.
`Tele2`: `line11(A)` ← `f132(A)` [32,0] `f132(A):-ext_number(A,B), prefix(B,[8,5,1,3,1,1,1,1]).` Calls received from number 85131111. *Expert's explanation:* The caller is the secretary's husband. She does not have a direct-access line, thus this call is transferred by an operator. *Expert's evaluation*: Novel information. *Remark.* Although the last literal formally identifies a *prefix* of the calling number, it is in fact the complete number of the caller.
`Tele3`: `line21(A)` ← `f54(A)` [81,254] `f54(A):-ext_number(A,B),prefix(B,[0,4]).` Calls received from a number that starts with 04. *Expert's explanation:* Prefix 04 is too general (code covers a large area) to find an explanation. *Expert's evaluation:* Novel information. Uncertain.

`Tele4`: `line28(A)` ← `f7(A)` [22,11]
`f7(A):-call_date(A,B),call_time(A,C),`
`ext_number(A,D), prev_attempt(B,C,D,[2,8], last_hour, unavailable).`
Calls received from a caller who has in the last hour attempted to directly (not through an operator) reach line 28, which was unavailable.
Expert's explanation: It is plausible that people try line 28 as the second attempt when line 21 is unavailable. Subgroup probably mostly covers people with technical difficulties with a product sold by person on line 21.
Expert's evaluation: Novel information.

the maximum feature length to 8. In this case, using a minimum coverage of 20 instances yields 138 features.

5.4 Results of subgroup discovery

An example feature in the Mutagenesis domain is `f12(A):-atm(A,B),atm_chr (B,C),lteq_c(C,0.142)` expressing that a drug contains an atom with charge less or equal to 0.142, or `f31(A):-benzene(A,B),benzene(A,C), connected( C,B)`, expressing the presence of two connected benzene rings in the chemical. In telecommunications, an example feature is `f99(A):-ext_number(A,B),prefix( B,[0,4,0,7])`, meaning that the caller's number starts with 0407. Another feature is `f115(A):- call_date(A,B),call_time(A,C),ext_number(A,D), prev _attempt(B,C,D,[3,1], today,unavailable)`, meaning that the caller (of the current call) has today tried to reach line 31, which was unavailable.

With these features, we use the RSD rule induction algorithm with altered covering strategy and heuristic function to produce sets of subgroup-describing rules.

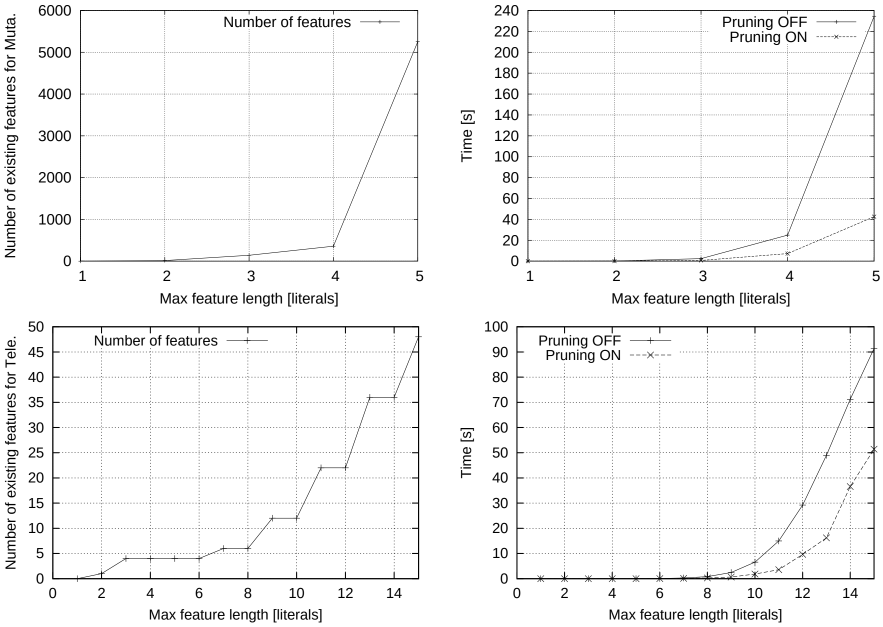

Fig. 2. The number of existing features (left) and the effect of undecomposability enabled pruning in the syntactic feature construction on efficiency (right) in the Mutagenesis (top) and Telecommunication (bottom) domain.

The characteristics of the discovered rules are shown in Table 2. Algo refers to the combination of search heuristic (A-accuracy, W-WRacc (weighted relative accuracy)) and covering algorithm (C-covering, W-WeCov (weighted covering with $\gamma = 1$)). S = significance, C = coverage, A = area under ROC curve. R : F = average number of rules/class : average number of features per rule. R$'$: F$'$ = same as above, only rules on covex hull are considered. The rule generation for a given class is terminated if the search space has been completely explored or 10 subgroup rules have been generated for that class in Telecommunication (5 in Mutagenesis). Reported results are averages and standard deviations from a 10-fold stratified cross-validation procedure.

The most significant observation about the results in Table 2 is that the *WRAcc* heuristic very significantly improves the performance with respect to the other accuracy heuristics, in terms of all three quality aspects.

Overall, the combination of *WRAcc* with the strategy of example weighting yields the best performance. This agrees with the findings in [17], where a more extensive empirical evaluation was conducted on a collection of (non-relational) subgroup-discovery problems, comparing the CN2 algorithm with CN2 incorporating the WRAcc heuristic, and further the CN2-SD system (which incorporates the *WRAcc* heuristic and the example weights). These three algorithms roughly

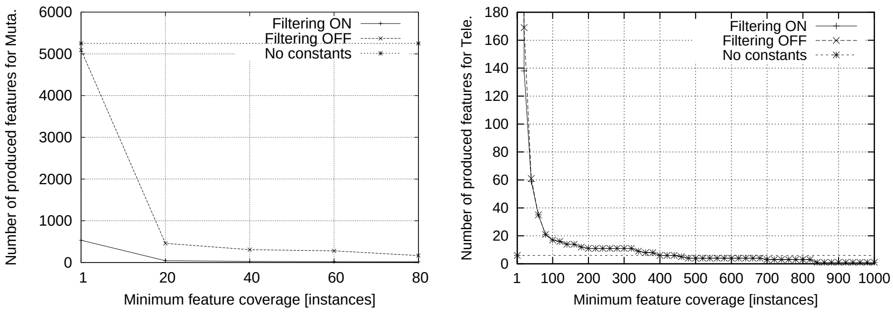

Fig. 3. The effect of using the constraint of unique and incomplete coverage ("Filtering ON" line) vs. ignoring this constraint ("Filtering OFF" line) and the user-adjustable minimum-coverage constrain (left-to-right decay) for Mutagenesis (left) and Telecommunication (right). The "No constants" curve (independent of the horizontal axis) corresponds to the number of features before instantiations to constants (before Step 2 of feature construction), this number is high in Mutagenesis due to many features inprovable with any instantiation to constants.

Table 2. Characteristics of subgroup-describing rules obtained by the RSD rule induction algorithm in the Mutagenesis and Telecom domains.

Mutagenesis

	Performance			**Complexity**	
Algo	S	C	A	R : F	R′ : F′
AC	1.99	11.33%	0.69	10.00 : 2.16	10.00 : 2.16
	(0.92)	(3.74)	(0.07)	(0.00 : 0.07)	(0.00 : 0.07)
AW	1.33	7.62%	0.58	10.00 : 2.50	10.00 : 2.50
	(1.05)	(4.88)	(0.06)	(0.00 : 0.11)	(0.00 : 0.11)
WC	4.22	35.81%	0.86	3.70 : 1.73	2.30 : 1.62
	(1.22)	(6.44)	(0.06)	(0.82 : 0.33)	(0.48 : 0.22)
WW	7.48	40.58%	0.90	10.00 : 2.63	6.50 : 2.43
	(1.28)	(4.74)	(0.04)	(0.00 : 0.07)	(0.97 : 0.11)

Telecommunication

	Performance			**Complexity**	
Algo	S	C	A	R : F	R′ : F′
AC	2.90	0.37%	0.55	7.36 : 2.39	6.88 : 2.47
	(0.38)	(0.05)	(0.02)	(0.12 : 0.04)	(0.19 : 0.04)
AW	2.25	0.25%	0.55	9.96 : 2.56	9.60 : 2.61
	(0.52)	(0.04)	(0.02)	(0.07 : 0.03)	(0.07 : 0.03)
WC	11.29	4.98%	0.67	6.12 : 2.17	5.20 : 2.28
	(1.71)	(0.54)	(0.02)	(0.16 : 0.04)	(0.16 : 0.04)
WW	11.99	4.02%	0.70	9.64 : 2.06	6.68 : 2.29
	(1.05)	(0.41)	(0.01)	(0.12 : 0.01)	(0.20 : 0.03)

correspond to the methods we denote above (in Table —refsd-results) as AC, WC, and WW, respectively. The combination of the accuracy heuristic with example weighting (AW) seems not to perform well in the domains considered.

6 Conclusions

This paper presents an approach to relational subgroup discovery, whose origins are based on the recent developments in subgroup discovery [33, 9] and propositionalization through first-order feature construction [8, 13, 18]. It presents the algorithm RSD which transforms a relational subgroup discovery problem to a propositional one, through efficiency-conscious first-order feature construction. Efficiency is boosted through the use of mode declarations and constraints used for pruning the search in the space of possible features.

Four variants of the RSD algorithm have been tested, by combining the standard accuracy search heuristic used in the construction of individual rules, with the standard covering algorithm used in the construction of a set of rules. The $WRAcc$ heuristic combined with the weighted covering algorithm is the preferred combination (due to an appropriate tradeoff between rule significance, coverage and complexity).

We have successfully applied the RSD algorithm in the Mutagenesis benchmark and the Telecom domain, a real-life dataset from a telecommunications company. These results have been evaluated as meaningful by the domain expert. Both the description of subgroups and their distributional characteristics make sense in many cases.

The idea of incrementally extending the feature set in dependence on the quality of the discovered subgroups, seems very much worth investigating in further work.

Acknowledgments

We are grateful to Peter Flach for the collaboration in the development of the CN2-SD algorithm and in the genesis phase of the RSD algorithm. We are grateful also to Jiří Zídek, Atlantis Telecom s.r.o. for the comments on the Telecom subgroups discovered. Nada Lavrač and Sašo Džeroski acknowledge the support of the Slovenian Ministry of Education, Science and Sport and the cInQ (Consortium on discovering knowledge with Inductive Queries) project, funded by the European Commission. Filip Železný[2]is supported by the DARPA EELD grant F30602-01-2-0571 and the Czech ministry of education grant MSM 21230013.

References

1. Rakesh Agrawal, Hiekki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in knowledge discovery and data mining*, pages 307–328, 1996.

2. John M. Aronis and Foster J. Provost. Efficiently constructing relational features from background knowledge for inductive machine learning. In *KDD Workshop*, 1994.
3. John M. Aronis, Foster J. Provost, and Bruce G. Buchanan. Eploiting background knowledge in automated discovery. In *KDD Workshop*, 1996.
4. R. Bayardo. Constraints in data mining. *SIGKDD Explorations*, 4(1), 2002.
5. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
6. L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 1–8. Springer-Verlag, 1998.
7. Tom Fawcett. Using rule sets to maximize ROC performance. In *ICDM*, pages 131–138, 2001.
8. P. Flach and N. Lachiche. 1BC: A first-order Bayesian classifier. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 92–103. Springer-Verlag, 1999.
9. D. Gamberger and N. Lavrač. Expert guided subgroup discovery: Methodology and application. *JAIR*, 17:501–527, 2002.
10. M. Garofalakis and R. Rastogi. Scalable data mining with model constraints. *SIGKDD Explorations*, 2(2):39–48, 2000.
11. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
12. W. Kloesgen. Explora: A multipattern and multistrategy discovery assistant, 1996.
13. Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In Saso Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, September 2001.
14. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
15. N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In: S. Džeroski and P. Flach (eds.): *Proceedings of the 9th International Workshop on Inductive Logic Programming*, 174–185, Springer-Verlag, 1999.
16. Nada Lavrač, Peter Flach, Brank Kavšek, and Ljupčo Todorovski. Adapting classification rule induction to subgroup discovery. In V. Kumar et al., editor, *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 266–273. IEEE Computer Society, December 2002.
17. Nada Lavrač,Brank Kavšek, Peter Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 2004 (in press).
18. Nada Lavrač and Peter A. Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, 2(4):458–494, October 2001.
19. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
20. D. Michie, S. Muggleton, D. Page, and A. Srinivasan. To the international computing community: A new East-West challenge. Technical report, Oxford University Computing laboratory, Oxford, UK, 1994.
21. S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.
22. Foster J. Provost and Tom Fawcett. Robust classification systems for imprecise environments. In *AAAI/IAAI*, pages 706–713, 1998.

23. L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.
24. Luc De Raedt, Hendrik Blockeel, Luc Dehaspe, and Wim Van Laer. Three companions for data mining in first order logic. In Saso Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.
25. G. Piatetsky-Shapiro and C. Matheus. The interestingness of deviation. In: *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 25–36, 1994.
26. A. Silberschatz and A. Tuzhilin. On Subjective Measures of Interestingness in Knowledge Discovery. In: *Knowledge Discovery and Data Mining*, 275–281, 1995.
27. A. Srinivasan and R.D. King. Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, volume 1314 of *Lecture Notes in Artificial Intelligence*, pages 89–104. Springer-Verlag, 1996.
28. A. Srinivasan, S. 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), 1996.
29. Ljupčo Todorovski, Peter Flach, and Nada Lavrač. Predictive performance of weighted relative accuracy. In Djamel A. Zighed, Jan Komorowski, and Jan Zytkow, editors, *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, pages 255–264. Springer-Verlag, September 2000.
30. F. Železný, P. Miksovsky, O. Stepankova, and J. Zidek. ILP for automated telephony. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 276–286, 2000.
31. F. Železný, J. Zidek, and O. Stepankova. A learning system for decision support in telecommunications. In *Proc. of the 1st Int. Conf. on Computing in an Imperfect World, Belfast 4/2002*. Springer-Verlag, 2002.
32. Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In Jan Komorowski and Jan Zytkow, editors, *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, Berlin, 1997. Springer Verlag.
33. Stefan Wrobel. Inductive logic programming for knowledge discovery in databases. In Saso Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 74–101. Springer-Verlag, September 2001.
34. Stefan Wrobel and Saso Džeroski. The ILP description learning problem: Towards a general model-level definition of data mining in ILP. In K. Morik and J. Herrmann, editors, *Proc. Fachgruppentreffen Maschinelles Lernen (FGML-95)*, 44221 Dortmund, 1995. Univ. Dortmund.