RSD: Relational subgroup discovery through first-order feature construction

Nada Lavrač[1], Filip Železný[2], and Peter Flach[3]

[1] Institute Jožef Stefan, Ljubljana, Slovenia
`nada.lavrac@ijs.si`
[2] Czech Technical University, Prague, Czech Republic
`zelezny@fel.cvut.cz`
[3] University of Bristol, Bristol, UK
`peter.flach@bristol.ac.uk`

Abstract. Relational rule learning is typically used in solving classification and prediction tasks. However, relational rule learning can be adapted also to subgroup discovery. This paper proposes a propositionalization approach to relational subgroup discovery, achieved through appropriately adapting rule learning and first-order feature construction. The proposed approach, applicable to subgroup discovery in individual-centered domains, was successfully applied to two standard ILP problems (East-West trains and KRK) and a real-life telecommunications application.

1 Introduction

Developments in *descriptive induction* have recently gained much attention.These involve mining of association rules (e.g., the APRIORI association rule learning algorithm [1]), subgroup discovery (e.g., the MIDOS subgroup discovery algorithm [22]), symbolic clustering and other approaches to non-classificatory induction.

The methodology presented in this paper can be applied to relational subgroup discovery. As in the MIDOS approach, a subgroup discovery task can be defined as follows: given a population of individuals and a property of those individuals 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. This paper aims at solving a slightly modified subgroup discovery task that can be stated as follows. Again, the input is a population of individuals and a property of those individuals we are interested in, and the output are population subgroups that are statistically 'most interesting': are as large as possible, have the most unusual statistical (distributional) characteristics with respect to the property of interest and are sufficiently distinct for detecting most of the target population.

Notice an important aspect of the above two definitions. In both, there is a predefined property of interest, meaning that both aim at characterizing popu-

lation subgroups of a given *target* class. This property indicates that rule learning may be an appropriate approach for solving the task. However, we argue that standard propositional rule learning [6, 16] and relational rule learning algorithms [19] are unsuitable for subgroup discovery. The main drawback is the use of the covering algorithm for rule set construction. Only the first few rules induced by a covering algorithm may be of interest as subgroup descriptions with sufficient coverage, thus representing a 'chunk' of knowledge characterizing a sufficiently large population of covered examples. Subsequent rules are induced from smaller and strongly biased example subsets, i.e., subsets including only positive examples not covered by previously induced rules. This bias prevents a covering algorithm to induce descriptions uncovering significant subgroup properties of the entire population. A remedy to this problem is the use of a weighted covering algorithm, as demonstrated in this paper, where subsequently induced rules with high coverage allow for discovering interesting subgroup properties of the entire population.

This paper investigates how to adapt classification rule learning approaches to subgroup discovery, by exploiting the information about class membership in training examples. This paper shows how this can be achieved by appropriately modifying the covering algorithm (weighted covering algorithm) and the search heuristics (weighted relative accuracy heuristic). The main advantage of the proposed approach is that each rule with 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 paper is organized as follows. In Section 2 the background for this work is explained: propositionalization through first-order feature construction, irrelevant feature elimination, the standard covering algorithm used in rule induction, the standard heuristics as well as the weighted relative accuracy heuristic, probabilistic classification and rule evaluation in the ROC space. Section 3 presents the proposed relational subgroup discovery algorithm. Section 4 presents the experimental evaluation in two standard ILP problems (East-West trains and KRK) and a real-life telecommunications application. Section 5 concludes by summarizing the results and presenting plans for further work.

2 Background

This section presents the backgrounds: propositionalization through first-order feature construction, irrelevant feature elimination, the standard covering algorithm used in rule induction, the standard heuristics as well as the weighted relative accuracy heuristic, probabilistic classification and rule evaluation in the ROC space.

2.1 Propositionalization through First-order Feature Construction

The background knowledge used to construct hypotheses is a distinctive feature of relational rule learning (and inductive logic programming, in general). It is

well known that relevant background knowledge may substantially improve the results of learning in terms of accuracy, efficiency, and the explanatory potential of the induced knowledge. On the other hand, irrelevant background knowledge will have just the opposite effect. Consequently, much of the art of inductive logic programming lies in the appropriate selection and formulation of background knowledge to be used by the selected ILP learner.

By devoting enough effort to the construction of features, to be used as background knowledge in learning, even complex relational learning tasks can be solved by simple propositional rule learning systems. In propositional learning, the idea of augmenting an existing set of attributes with new ones is known under the term constructive induction. A first-order counterpart of constructive induction is predicate invention. This work takes the middle ground: we perform a simple form of predicate invention through first-order feature construction, and use the constructed features for relational rule learning, which thus becomes propositional.

Our approach to first-order feature construction can be applied in the so-called *individual-centered* domains, where there is a clear notion of individual, and learning occurs at the level of individuals only. Such domains include classification problems in molecular biology, for example, where the individuals are molecules. Often, individuals are represented by a single variable, and the target predicates are either unary predicates concerning boolean properties of individuals, or binary predicates assigning an attribute-value or a class-value to each individual. It is however also possible that individuals are represented by tuples of variables.

In our approach to first-order feature construction, described in [9, 12, 10], local variables referring to parts of individuals are introduced by so-called *structural predicates*. The only place where nondeterminacy can occur in individual-centered representations is in structural predicates. Structural predicates introduce new variables. 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 [13] (called *properties* in [9]) 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.

Individual-centered representations have the advantage of a strong language bias, because local variables in the bodies of rules either refer to the individual or to parts of it. However, not all domains are amenable to the approach we use in this paper - in particular, we cannot learn recursive clauses, and we cannot deal with domains where there is not a clear notion of individual (e.g., many program synthesis problems).

2.2 Irrelevant Feature Elimination

Let L denote the set of all features constructed by a first-order feature construction algorithm. Some features defined by the language bias may be irrelevant for the given learning task. Irrelevant features can be detected and eliminated in

preprocessing. Besides reducing the hypothesis space and facilitating the search for the solution, the elimination of irrelevant features may contribute to a better understanding of the problem domain.

It can be shown that if a feature $l' \in L$ is irrelevant then for every complete and consistent hypothesis $H = H(E, L)$, built from example set E and feature set L whose description includes feature l', there exists a complete and consistent hypothesis $H' = H(E, L')$, built from the feature set $L' = L \backslash \{l'\}$ that excludes l'. This theorem is the basis of an irrelevant feature elimination algorithm proposed in [15].

Note that usually the term feature is used to denote a positive literal (or a conjunction of positive literals; let us, for the simplicity of the arguments below, assume that a feature is a single positive literal). In the hypothesis language, the existence of one feature implies the existence of two complementary literals: a positive and a negated literal. Since each feature implies the existence of two literals, the necessary and sufficient condition for a feature to be eliminated as irrelevant is that both of its literals are irrelevant.

This observation directly implies the approach taken in this work. First we convert the starting feature set to the corresponding literal set which has twice as many elements. After that, we eliminate the irrelevant literals and, in the third step, we construct the reduced set of features which includes all the features which have at least one of their literals in the reduced literal set.

It must be noted that direct detection of irrelevant features (without conversion to and from the literal form) is not possible except in the trivial case where two (or more) features have identical values for all training examples. Only in this case a feature f exists whose literals f and $\neg f$ cover both literals g and $\neg g$ of some other feature. In a general case if a literal of feature f covers some literal of feature g then the other literal of feature g is not covered by the other literal of feature f. But it can happen that this other literal of feature g is covered by a literal of some other feature h. This means that although there is no such feature f that covers both literals of feature g, feature g can still turn out to be irrelevant.

2.3   Rule Induction Using the Covering Algorithm

Rule learning typically consists of two main procedures: the search procedure that performs search in order to find a single rule and the control procedure that repeatedly executes the search. In the propositional rule learner CN2 [5, 6], for instance, the search procedure performs beam search using classification accuracy of the rule as a heuristic function. The accuracy of rule $H \leftarrow B$ is equal to the conditional probability of head H, given that the body B is satisfied: $Acc(H \leftarrow B) = p(H|B)$.

The accuracy measure can be replaced by the weighted relative accuracy, defined in Equation 1. Furthermore, different probability estimates, like the Laplace [4] or the m-estimate [3, 7], can be used for estimating the above probability and the probabilities in Equation 1.

Additionally, a rule learner can apply a significance test to the induced rule. The rule is considered to be significant, if it locates a regularity unlikely to have occurred by chance. To test significance, for instance, CN2 uses the likelihood ratio statistic [6] that measures the difference between the class probability distribution in the set of examples covered by the rule and the class probability distribution in the set of all training examples. Empirical evaluation in [4] shows that applying a significance test reduces the number of induced rules (but also slightly degrades the predictive accuracy).

Two different control procedures are used in CN2: one for inducing an ordered list of rules and the other for the unordered case. When inducing an ordered list of rules, the search procedure looks for the best rule, according to the heuristic measure, in the current set of training examples. The rule predicts the most frequent class in the set of examples, covered by the induced rule. Before starting another search iteration, all examples covered by the induced rule are removed. The control procedure invokes a new search, until all the examples are covered.

In the unordered case, the control procedure is iterated, inducing rules for each class in turn. For each induced rule, only covered examples belonging to that class are removed, instead of removing all covered examples, like in the ordered case. The negative training examples (i.e., examples that belong to other classes) remain and positives are removed in order to prevent CN2 finding the same rule again.

2.4 The Weighted Relative Accuracy Heuristic

Weighted relative accuracy can be meaningfully applied both in the descriptive and predictive induction framework; in this paper we apply this heuristic for subgroup discovery.

We use the following notation. Let $n(B)$ stand for the number of instances covered by rule $H \leftarrow B$, $n(H)$ stand for the number of examples of class H, and $n(H.B)$ stand for the number of correctly classified examples (true positives). We use $p(H.B)$ etc. for the corresponding probabilities. We then have that rule accuracy can be expressed as $Acc(H \leftarrow B) = p(H|B) = \frac{p(H.B)}{p(B)}$. Weighted relative accuracy [14, 21], a reformulation of one of the heuristics used in MIDOS [22], is defined as follows.

$$WRAcc(H \leftarrow B) = p(B).(p(H|B) - p(H)). \tag{1}$$

Weighted relative accuracy consists of two components: generality $p(B)$, and relative accuracy $p(H|B) - p(H)$. The second term, relative accuracy, is the accuracy gain relative to the fixed rule $H \leftarrow true$. The latter rule predicts all instances to satisfy H; a rule is only interesting if it improves upon this 'default' accuracy. Another way of viewing relative accuracy is that it measures the utility of connecting rule body B with a given rule head H. However, it is easy to obtain high relative accuracy with highly specific rules, i.e., rules with low generality $p(B)$. To this end, generality is used as a 'weight', so that weighted relative

accuracy trades off generality of the rule ($p(B)$, i.e., rule coverage) and relative accuracy ($p(H|B) - p(H)$).

2.5 Probabilistic Classification

The induced rules can be ordered or unordered. Ordered rules are interpreted as a decision list [20] in a straight-forward manner: when classifying a new example, the rules are sequentially tried and the first rule that covers the example is used for prediction.

In the case of unordered rule sets, the distribution of covered training examples among classes is attached to each rule. Rules of the form:

$$H \leftarrow B\ [ClassDistribution]$$

are induced, where numbers in the $ClassDistribution$ list denote, for each individual class H, how many training examples of this class are covered by the rule. When classifying a new example, all rules are tried and those covering the example are collected. If a clash occurs (several rules with different class predictions cover the example), a voting mechanism is used to obtain the final prediction: the class distributions attached to the rules are summed to determine the most probable class.

2.6 Area Under the ROC Curve Evaluation

A point on the $ROC\ curve$[1] shows classifier performance in terms of false alarm or $false\ positive\ rate\ FPr = \frac{FP}{TN+FP}$ (plotted on the X-axis) that needs to be minimized, and sensitivity or $true\ positive\ rate\ TPr = \frac{TP}{TP+FN}$ (plotted on the Y-axis) that needs to be maximized. In the ROC space, an appropriate tradeoff, determined by the expert, can be achieved by applying different algorithms, as well as by different parameter settings of a selected data mining algorithm.

3 Relational Subgroup Discovery

We have devised a relational subgroup discovery system RSD on principles that employ the following main ingredients: exhaustive first-order feature construction, elimination of irrelevant features, implementation of a relational rule learner, use of the weighted covering algorithm and incorporation of example weights into the weighted relative accuracy heuristic.

The input to RSD consists of

- a relational database (further called *input data*) 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, and
- a mode-language definition used to construct first-order features.

---

[1] ROC stands for Receiver Operating Characteristic [11, 18]

The main output of RSD is a set of subgroups whose class-distributions differ substantially from those of the complete data-set. The subgroups are identified by conjunctions of symbols of pre-generated first-order features. As a by-product, RSD also provides a file containing the mentioned set of features and offers to export a single relation (as a text file) with rows corresponding to individuals and fields containing the truth values of respective features for the given individual. This table is thus a propositionalized representation of the input data and can be used as an input to various attribute-value learners.

3.1 RSD First-Order Feature Construction

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

- Identifying all first-order literal conjunctions that by definition form a feature (see [9], briefly described in Section 2.1), 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.
- 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.
- Detecting irrelevant features (see [15], briefly described in Section 2.2) and generating propositionalized representations of the input data using the generated feature set.

Identifying features. Motivated by the need to easily recycle language-bias declarations already present for numerous ILP problems, RSD accepts declarations very similar to those used by the systems Aleph [2] or Progol [17], including variable typing, moding, setting a *recall* parameter etc, used to syntactically constrain the set of possible features. For example, a structural predicate declaration in the well-known domain of East-West trains would be such as

```
:-modeb(1, hasCar(+train, -car)).
```

where the recall number 1 determines that a feature can address at most one car of a given train. Property predicates are those with no output variables (labeled with the minus sign). The head declaration always contains exactly one variable of the input mode (e.g. +train in our example).

Various settings such as the maximal *feature length* denoting the maximal number of literals allowed in a feature, maximal *variable depth* etc. can also be specified, otherwise their default value is assigned.

RSD will produce the exhaustive set of features satisfying the mode and setting declarations. No feature produced by RSD can be decomposed into a conjunction of two features.

Employing constants. As opposed to Aleph or Progol declarations, RSD does not use the # sign to denote an argument which should be a constant. In the mentioned systems, constants are provided by a single saturated example, while RSD extract constants from the whole input data. The user can instead utilize the reserved property predicate `instantiate/1`, which does not occur in the background knowledge, to specify a variable which should be substituted with a constant. For example, out of the following declarations

```
:-modeh(1, train(+train)).
:-modeb(1, hasCar(+train, -car)).
:-modeb(1, hasLoad(+car, -load)).
:-modeb(1, hasShape(+load, -shape).
:-modeb(3, instantiate(+shape)).
```

exactly one feature will be generated:[2]

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

However, in the second step, after consulting the input data, `f1` will be substituted by 3 features (due to the recall parameter 3 in the last declaration) where the `instantiate/1` literal is removed and the `D` variable is substituted with 3 most frequent constants out of all values of `D` which make the body of `f1` provable in the input data. One of them will be

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

Filtering and applying features. RSD implements the simplest scheme of feature filtering from [15]. This means that features with complete of empty coverage on the input data will be retracted from the feature set. In the current implementation, RSD does not check for irrelevance of a feature caused by the presence of other features. As the product of this third, final step of feature construction, the system exports an attribute representation of the input data based on the truth values of respective features, in a file of parametrizable format.

3.2 RSD Rule Induction Algorithm

A part of RSD is a subgroup discovery program which can accept data propositionalized by the feature constructor described above. The algorithm acquires some basic principles of the CN2 rule learner [6], which are however adapted in several substantial ways to meet the interests of subgroup discovery. The principal modifications are outlined below.

---

[2] Strictly speaking, the feature is solely the *body* of the listed clause. However, clauses such as the one listed will also be called features in the following as this will cause no confusion.

The Weighted Covering Algorithm. In the classical covering algorithm for rule set induction, only the first few induced rules may be of interest as subgroup descriptors with sufficient coverage, since subsequently induced rules are induced from biased example subsets, i.e., subsets including only positive examples not covered by previously induced rules. This bias constrains the population for subgroup discovery in a way that is unnatural for the subgroup discovery process which is, in general, aimed at discovering interesting properties of subgroups of the entire population. In contrast, the subsequent rules induced by the proposed weighted covering algorithm allow for discovering interesting subgroup properties of the entire population.

The weighted covering algorithm performs in such a way that covered positive examples are not deleted from the current training set. Instead, in each run of the covering loop, the algorithm stores with each example a count how many times (with how many rules induced so far) the example has been covered. Weights derived from these example counts then appear in the computation of *WRAcc*. Initial weights of all positive examples e_j equals $w(e_j, 0) = 1$. Weights of covered positive examples decrease according to the formula $w(e_j, i) = \frac{1}{i+1}$, where $w(e_j, i)$ is the weight of example e_j being covered i times.[3]

Modified WRAcc Heuristic with Example Weights. The modification of CN2 reported in [21] affected only the heuristic function: weighted relative accuracy was used as search heuristic, instead of the accuracy heuristic of the original CN2, while everything else stayed the same. In this work, the heuristic function was further modified to enable handling example weights, which provide the means to consider different parts of the instance space in each iteration of the weighted covering algorithm.

In the *WRAcc* computation (Equation 1) all probabilities are computed by relative frequencies. An example weight measures how important it is to cover this example in the next iteration. The initial example weight $w(e_j, 0) = 1$ means that the example hasn't been covered by any rule, meaning 'please cover this example, it hasn't been covered before', while lower weights mean 'don't try too hard on this example'. The modified *WRAcc* measure is then defined as follows

$$WRAcc(H \leftarrow B) = \frac{n'(B)}{N'}\left(\frac{n'(H.B)}{n'(B)} - \frac{n'(H)}{N'}\right). \qquad (2)$$

where N' is the sum of the weights of all examples, $n'(B)$ is the sum of the weights of all covered examples, and $n'(H.B)$ is the sum of the weights of all correctly covered examples.

To add a rule to the generated rule set, the rule with the maximum *WRAcc* measure is chosen out of those rules in the search space, which are not yet present

---

[3] Whereas this approach is referred to as *additive*, another option is the *multiplicative* approach, where, for a given parameter $\gamma < 1$, weights of covered examples decrease as follows: $w(e_j, i) = \gamma^i$. Note that the weighted covering algorithm with $\gamma = 1$ would result in finding the same rule over and over again, whereas with $\gamma = 0$ the algorithm would perform the same as the standard CN2 algorithm.

in the rule set produced so far (all rules in the final rule set are thus distinct, without duplicates).

Probabilistic classification and Area Under the ROC curve evaluation. The method, which is used for evaluation of predictive performance of the subgroup discovery results employs the combined probabilistic classifications of all subgroups (a set of rules as a whole). If we always choose the most likely predicted class, this corresponds to setting a fixed threshold 0.5 on the positive probability: if the positive probability is larger than this threshold we predict positive, else negative. A ROC curve can be constructed by varying this threshold from 1 (all predictions negative, corresponding to (0,0) in the ROC space) to 0 (all predictions positive, corresponding to (1,1) in the ROC space). This results in $n+1$ points in the ROC space, where n is the total number of examples. Equivalently, we can order all examples by decreasing the predicted probability of being positive, and tracing the ROC curve by starting in (0,0), stepping up when the example is actually positive and stepping to the right when it is negative, until we reach (1,1).[4] The area under this ROC curve indicates the combined quality of all subgroups. This method can be used with a test set or in cross-validation: the resulting curve is not necessarily convex, but the area under the curve still indicates the quality of the combined subgroups for probabilistic prediction.[5]

4 Experimental evaluation

4.1 Materials

We have performed experiments with RSD on two popular ILP data sets: the King-Rook-King illegal chess endgame positions (KRK) and East-West trains.

We applied RSD also to a real-life problem in telecommunications. The data (described in detail in [23]) represent incoming calls to an enterprise, which were transferred to a particular person by the telephone receptionist. The company has two rather independent divisions ('datacomm', 'telecomm') and people in each of them may have a technical or commercial role. These two binary divisions define four classes of incoming calls ('data_tech', 'data_comm', 'tele_tech', 'tele_comm'), depending on the person the call was redirected to. The fifth class ('other') labels the calls (of smaller interest) going to other employees than those mentioned. The problem is to define subgroups of incoming calls usually falling into a given class. The motivation for subgroup discovery in this domain is to select people within a given class that can substitute each other in helping the caller.

Table 1 lists the basic properties of the experimental data.

---

[4] In the case of ties, we make the appropriate number of steps up and to the right at once, drawing a diagonal line segment.

[5] A description of this method applied to decision tree induction can be found in [8].

Domain	Individual	No. of examples	No. of classes
KRK	KRK position	1000	2
Trains	Train	20	2
Telecommunication	Incoming call	1995	5

Table 1. Basic properties of the experimental data.

4.2 Procedures

We have applied the RSD algorithm with each of the data sets in the following manner.

– First, we generated a set of features and, except for the KRK domain, we expanded the set with features containing variable instantiations.
– Then the feature sets were used to produce a propositional representation of each of the data sets.
– From propositional data we induced rules with the RSD subgroup discovery algorithm as well as with the standard coverage approach.
– Finally, we compared the pairs of rule sets in terms of properties that are of interest for subgroup discovery.
– In one domain we evaluated the induced rule sets also from the point of view of predictive classification. To do so, we performed a classification test on the unseen part of the data, where rules were interpreted by the method of probabilistic classification, suing a ROC plot to evaluate the results.

4.3 Results

Feature construction. Table 2 shows the values of the basic parameters of the language-bias declarations provided to the feature constructor for each given domain as well as the number of generated features before and after the expansion due to variable instantiations.

	KRK	Trains	Tele
No. of body declarations	10	16	11
Max. clause lenght	3	8	8
Max var. depth	2	4	4
No. of features before instantiations	42	219	4
No. of features after instantiations	42	2429	3729

Table 2. The language bias and number of generated features for each of the experimental domain.

Table 3 below presents a few examples of features generated for each of the three domains together with a verbal interpretation.

KRK	`f6(A):-king1_rank(A,B),rook_rank(A,C),adj(C,B).`
meaning	first king's rank adjacent to rook's rank
Trains	`f5(A):-hasCar(A,B),carshape(B,ellipse),carlength(B,short).`
meaning	has a short elliptic car
Tele	`f3(A):-call_date(A,B),day_is(B,mon).`
meaning	call accepted on Monday
	`f301(A):-ext_number(A,B),prefix(B,[0,4,0,7]).`
meaning	caller's number starts with 0407

Table 3. Examples of generated features.

Subgroup discovery. The following experiment compares the results of the search for interesting subgroups by means of four techniques described in detail in the previous sections:

- Standard covering algorithm with rule accuracy as a selection measure (Cov + Acc).
- Example weighting (i.e., weighted covering algorithm) with rule accuracy as a selection measure (We + Acc).
- Standard covering algorithm with weighted relative accuracy as a selection measure (Cov + WRAcc).
- Example weighting with weighted relative accuracy as a selection measure (We + WRAcc).

The resulting rule sets are evaluated on the basis of average rule *significance* and average rule coverage. The significance of a rule is measured as in the CN2 algorithm [6], i.e., as the value of

$$\sum_i f_i \log \frac{f_i}{p_i} \tag{3}$$

where for each class i, p_i denotes the number of instances classified into i within the whole data set, and f_i denotes the number of such classified instances in the subset, where the rule's body holds true.

Table 4 shows the average significance and coverage values of rules produced in the three respective domains by each of the four combination of techniques, while Figure 1 shows the significance values for all individual rules.

By making pairwise comparisons we can investigate the influence of replacing the standard covering algorithm (*Cov*) by the weighted covering algorithm *We*), and the affects of replacing the standard accuracy heuristic (*Acc*) with the weighted relative accuracy heuristic (*WRAcc*). The following observations can be made from the quantitative results.

- *Example weighting increases* the average rule *significance* with respect to the standard covering algorithm for both cases of combination with the accuracy
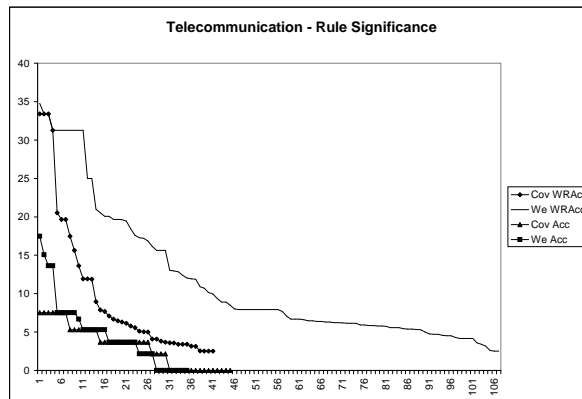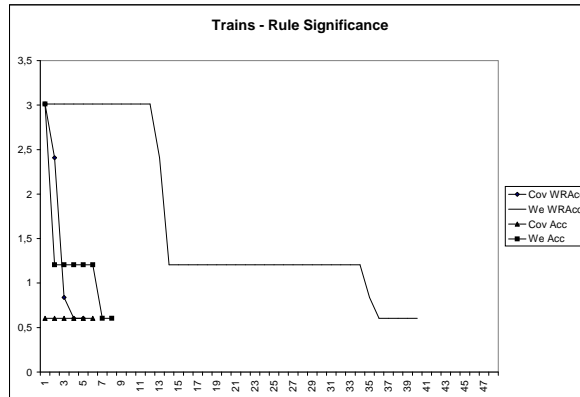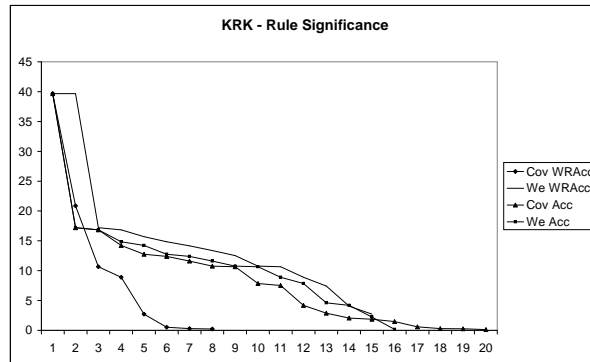
Fig. 1. The significance of individual rules produced by each of the four combinations of techniques in the three data domains. Rules are sorted decreasingly by their significance value. Note that the number of rules produced by the example weighting algorithm (*We*) is parameterizable by the setting of the weight-decay rate and the total weight threshold for stopping the search. We used the same threshold for all *We*-based experiments.

KRK	Cov + WRAcc	We + WRAcc	Cov + Acc	We + Acc
Avg. Significance	10.48	15.23	8.76	11.80
Avg. Coverage	16.84%	17.11%	8.56%	14.97%

Trains	Cov + WRAcc	We + WRAcc	Cov + Acc	We + Acc
Avg. Significance	1.49	1.69	0.60	1.28
Avg. Coverage	26.66%	17.00%	6.67%	14.16%

Tele	Cov + WRAcc	We + WRAcc	Cov + Acc	We + Acc
Avg. Significance	9.92	11.55	3.17	4.87
Avg. Coverage	5.78%	3.2%	0.0012%	0.0012%

Table 4. Average values of significance and coverage of the resulting rules of subgroup discovery conducted by four different combinations of techniques.

or weighted relative accuracy measure, in all three data domains. This follows from Table 4 above as well as from Figure 1.

– The use of *weighted relative accuracy* heuristic *increases* the average rule *coverage* with respect to the case of using the accuracy measure. This holds for both cases of combination with the standard covering algorithm or the technique of example weighting, in all three data domains. This follows from Table 4 (it is particularly striking in the Telecommunication domain).

– A consequence of the previous observation is that in the standard covering algorithm, the use of *weighted relative accuracy* leads to smaller output rule sets. This can be verified in Figure 1.

– It may seem surprising that the combination Cov + WRAcc yields a higher average rule coverage than We + WRAcc for Trains and Telecommunication. However, note that it does not make sense to compare coverage results across Cov/We. The reason is that the rule generation process using We (gradually tending to produce more specific rules) is stopped at an instant dictated by the user-specified threshold parameter for the sum of all examples' weights. Thus the coverage values are to be compared between methods both using Cov, and between methods both using We with the same threshold value setting.

Classification. Although the aim of RSD is to merely discover rules representing interesting subgroups, such rules may as well be used for classification purposes. For a binary classification problem, we can interpret the rule set as explained in Section 3.2 to obtain a probabilistic classifier which is then evaluated by means of the area under the ROC curve value. For the two binary-class KRK data set we have thus repeated the rule induction process with only a part of the propositionalized data (750 examples in KRK) and compare RSD rules with the standard covering algorithm.

We have made the comparisons of the performance of the four methods in terms of the area under the ROC curve only in the KRK domain. The reasons for this decision are that

- the Trains domain is too small for splitting the training examples into a training set and a separate test set, and
- the Telecommunications domain is a multi-class problem where the area under ROC curve elevation can not be applied in a straight-forward way.

Table 5 lists the area under ROC values achieved by each of the four combinations of methods on the KRK domain.[6] The values have been obtained by using the subgroup description rules for probabilistic classification as described in Section 3.2. The corresponding ROC curves are shown in Figure 2. It can be observed the using the WRAcc heuristic improved the predictive performance both with the standard coverage approach and the example weighting approach. We also observe that using example weighting yielded smaller predictive accuracies than standard coverage in both combinations with WRAcc and Acc. Our explanation is that due to the chosen parameterization of slow weight decrease and low total weight threshold we obtained an overly complex predictive model, composed of too many rules, leading to poorer predictive performance. The chosen parameterization was motivated by the aim of obtaining an exhaustive description of possibly interesting subgroups, but alternative, prediction-aimed settings as well as filtering based on significance measures are part of the future experimentation.

KRK	Cov + WRAcc	We + WRAcc	Cov + Acc	We + Acc
Area Under ROC	0.93	0.84	0.92	0.80

Table 5. The area under ROC values for 4 possible combinations of methods on the KRK domain.

5 Conclusions

We have developed the system RSD which discovers interesting subgroups in classified relational data. The strategy followed by the system begins with converting the relational data into a single relation via first-order feature construction. RSD can 'sensitively' extract constants from the input data and employ them in the feature definitions. It can also detect and remove some irrelevant features, but so far only in a relatively simplified way. Finally, to identify interesting subgroups, RSD takes advantage of the method of example-weighting

---

[6] Note that we have used a separate test set for evaluating the results - the more elaborate evaluation using cross-validation is left for further work.
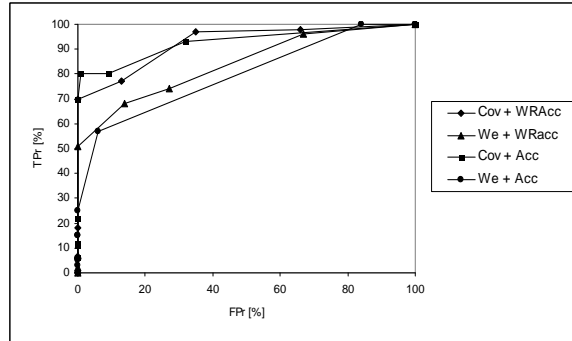
Fig. 2. The ROC curves for 4 possible combinations of methods on the KRK domain.

(used in the so-called weighted covering algorithm) and employing the *WRAcc* measure as a heuristic for rule evaluation.

In three experimental domains, we have shown that example weighting improves the average significance of discovered rules with respect to the standard covering algorithm. Also, the use of weighted relative accuracy increases the average coverage (generality) of resulting rules. The combined effect of the two techniques is thus an important contribution to the problem of subgroup discovery.

From the classification point of view, we observed that the WRAcc heuristic improved the predictive performance on the tested domain compared to the simple accuracy measure heuristic. On the other hand, example-weighting yielded a smaller predictive performance compared to the standard coverage approach. This is in our opinion due to the chosen parameterization of the algorithm motivated by the need of exhaustive description. In future work we will use test the results using cross-validation. We will also experiment with alternative, prediction-aimed settings.

Acknowledgements

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetski-Shapiro, P. Smyth and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, 307–328. AAAI Press, 1996.

2. A. Srinivasan and R.D. King. Feature construction with Inductive Logic Programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37–57, 1999.

3. B. Cestnik. Estimating probabilities: A crucial task in machine learning. In L. Aiello, editor, *Proc. of the 9th European Conference on Artificial Intelligence*, 147–149. Pitman, 1990.

4. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In Y. Kodratoff, editor, *Proc. of the 5th European Working Session on Learning*, 151–163. Springer, 1989.

5. P. Clark and T. Niblett. Induction in noisy domains. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning (Proc. of the 2nd European Working Session on Learning)*, 11–30. Sigma Press, 1987.

6. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

7. S. Džeroski, B. Cestnik, and I. Petrovski. (1993) Using the m-estimate in rule induction. *Journal of Computing and Information Technology*, 1(1):37 – 46, 1993.

8. C. Ferri-Ramírez, P.A. Flach, and J. Hernandez-Orallo. Learning decision trees using the area under the ROC curve. In *Proc. of the 19th International Conference on Machine Learning*, 139–146. Morgan Kaufmann, 2002.

9. P.A. Flach and N. Lachiche. 1BC: A first-order Bayesian classifier. In *Proc. of the 9th International Workshop on Inductive Logic Programming*, 92–103. Springer, 1999.

10. S. Kramer, N. Lavrač and P.A. Flach. Propositionalization approaches to relational data mining. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, 262–291. Springer, 2001.

11. M. Kukar, I. Kononenko, C. Grošelj, K. Kralj, and J.J. Fettich. Analysing and improving the diagnosis of ischaemic heart disease with machine learning. *Artificial Intelligence in Medicine*, special issue on *Data Mining Techniques and Applications in Medicine*, 16, 25–50. Elsevier, 1998.

12. N. Lavrač and P. Flach. An extended transformation approach to Inductive Logic Programming. *ACM Transactions on Computational Logic* 2(4): 458–494, 2001.

13. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.

14. N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *Proc. of the 9th International Workshop on Inductive Logic Programming*, 74–185. Springer, 1999.

15. N. Lavrač, D. Gamberger, and V. Jovanoski. (1999). A study of relevance for learning in deductive databases. *Journal of Logic Programming 40,* 2/3 (August/September), 215–249.

16. R.S. Michalski, I. Mozetič, J. Hong, and N. Lavrač. The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In *Proc. 5th National Conference on Artificial Intelligence*, 1041–1045. Morgan Kaufmann, 1986.

17. S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4): 245–286, 1995.
18. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3), 203–231, 2001.
19. J.R. Quinlan. Learning Logical definitions from relationa. *Machine Learning*, 5(3): 239–266, 1990.
20. R.L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.
21. L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In D.A. Zighed, J. Komorowski, and J. Zytkow, editors, *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, 255–264. Springer, 2000.
22. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proc. First European Symposium on Principles of Data Mining and Knowledge Discovery*, 78–87. Springer, 1997.
23. F. Železný, J. Zídek, and O. Štěpánková. A learning system for decision support in telecommunications. In *Proc. First International Conference on Computing in an Imperfect World*, 88–101. Springer, 2002.