Anita Valmarska, , Nada Lavrač, Johannes Fürnkranz, Marko Robnik-Šikonja: Refinement and selection heuristics in subgroup discovery and classification rule learning. Expert systems with applications, 81:147-162, 2017

Refinement and Selection Heuristics in Subgroup Discovery and Classification Rule Learning

Anita Valmarska[a,b,*], Nada Lavrač[a,b,c], Johannes Fürnkranz[d], Marko Robnik-Šikonja[e]

[a] *Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia*
[b] *Jožef Stefan International Postgraduate School, Jamova 39, Ljubljana, Slovenia*
[c] *University of Nova Gorica, Vipavska 13, Nova Gorica, Slovenia*
[d] *TU Darmstadt, Darmstadt, Germany*
[e] *University of Ljubljana, Faculty of Computer and Information Science, Slovenia*

Abstract

Classification rules and rules describing interesting subgroups are important components of descriptive machine learning. Rule learning algorithms typically proceed in two phases: rule refinement selects conditions for specializing the rule, and rule selection selects the final rule among several rule candidates. While most conventional algorithms use the same heuristic for guiding both phases, recent research indicates that the use of two separate heuristics is conceptually better justified, improves the coverage of positive examples, and may result in better classification accuracy. The paper presents and evaluates two new beam search rule learning algorithms: DoubleBeam-SD for subgroup discovery and DoubleBeam-RL for classification rule learning. The algorithms use two separate beams and can combine various heuristics for rule refinement and rule selection, which widens the search space and allows for finding rules with improved quality. In the classification rule learning setting, the experimental results confirm previously shown benefits of using two separate heuristics for rule refinement and rule selection. In subgroup discovery, DoubleBeam-SD algorithm variants outperform several state-of-the-art related algorithms.

Keywords: rule learning, subgroup discovery, inverted heuristics.

*Corresponding author.
Email addresses: anita.valmarska@ijs.si (Anita Valmarska), nada.lavrac@ijs.si (Nada Lavrač), fuernkranz@informatik.tu-darmstadt.de (Johannes Fürnkranz), marko.robnik@fri.uni-lj.si (Marko Robnik-Šikonja)

1. Introduction

While most data mining techniques aim at optimizing predictive performance of the induced models, their comprehensibility is of ultimate importance for expert systems and decision support. Examples of application areas in need of transparent models include medicine, law, finance and knowledge discovery (Bibal and Frénay, 2016).

Rule learning is a symbolic data analysis technique that can be used to construct understandable models or patterns describing the data (Michalski, 1969; Clark and Niblett, 1989; Fürnkranz et al., 2012). As one of the standard machine learning techniques it has been used in numerous applications. Compared to statistical learning techniques, the key advantage of rule learning is its simplicity and human understandable outputs. Therefore, the development of new rule learning algorithms for constructing understandable models and patterns is in the core interest of the data mining community.

Symbolic data analysis techniques can be divided into two categories. Techniques for *predictive induction* produce models, typically induced from labeled data, which are used to predict the label of previously unseen examples. The second category consists of techniques for *descriptive induction*, where the aim is to find comprehensible patterns, typically induced from unlabeled data. There are also descriptive induction techniques that learn descriptive rules from labeled data, which are referred to as *supervised descriptive rule discovery* techniques (Kralj Novak et al., 2009). Typical representatives of these techniques are subgroup discovery (SD) (Klösgen, 1996; Wrobel, 1997; Atzmueller, 2015), contrast set mining (CSM) (Bay and Pazzani, 2001), and emerging pattern mining (EPM) (Dong and Li, 1999) techniques. For instance, the task of subgroup discovery is to find interesting subgroups in the population, i.e. subgroups that have a significantly different class distribution than the entire population (Klösgen, 1996; Wrobel, 1997). The result of subgroup discovery is a set of individual rules, where the rule consequence is a class label.

An important characteristic of subgroup discovery is that its task is a combination of predictive and descriptive rule induction. It provides understandable descriptions of subgroups of individuals which share a common target property of interest. This feature of subgroup discovery has inspired many researchers to investigate new methods that will be more effective in finding interesting patterns in the data. Most subgroup discovery approaches build on classification algorithms, e.g., EXPLORA (Klösgen, 1996), MIDOS (Wrobel, 1997), SD (Gamberger and Lavrač, 2002), CN2-SD (Lavrač et al., 2004), and RSD (Lavrač et al., 2002), or on algorithms for association rule

learning, e.g., APRIORI-SD (Kavšek et al., 2003), SD-MAP (Atzmüller and Puppe, 2006), and Merge-SD (Grosskreutz and Rüping, 2009).

The main difference between classification rule learning and subgroup discovery is that subgroup discovery algorithms construct individual rules describing the properties of individual groups of target class instances, while classification rule learning algorithms construct a set of classification rules covering the entire problem space.

A common property of classification rule learning and subgroup discovery is that rule construction is performed in two phases: the rule refinement and the rule selection phase. Typically, different types of heuristics are used for classification rule induction and subgroup induction. Researchers usually choose one heuristic and use the same heuristic in the two phases of the rule construction process: (i) a heuristic is used to evaluate *rule refinements*, i.e. to select which of the refinements (specializations) of the current rule will be further explored, and (ii) the same heuristic is used in *rule selection* to decide which of the constructed rules will be added to the rule set. For learning classification rules, Stecher et al. (2014) proposed to use separate heuristics for each of the two rule construction phases, and suggested that in the refinement phase, so-called *inverted heuristics* should be used for evaluating the relative gain obtained by refining the current rule. The key idea of these heuristics is that while most conventional rule learning heuristics, such as the Laplace or the m-estimate, anchor their evaluation on the empty rule that does not cover any examples, inverted heuristics anchor the point of view on the base rule, which is more appropriate for a top-down refinement process.

In this paper, we test the utility of inverted heuristics in the context of subgroup discovery as well as in the context of classification rule learning. For this purpose we have developed two new beam search rule learning algorithms, named DoubleBeam-SD for subgroup discovery and DoubleBeam-RL for classification rule learning, respectively. The algorithms allow to combine various heuristics for rule refinement and rule selection, with the goal of determining their optimal combination, and, in consequence, learn rules with better coverage and better descriptive power without compromising rule accuracy. The introduction of two separate beams enlarges the search space, enabling the learner to find rule sets that are more accurate as well as more interesting to the end user. For example, physicians appreciate rules that are highly accurate when used in patient classification, but prefer understandable rules that precisely characterize the patients in terms of the features that distinguish the patients from the control group.

We compare the double beam search algorithms to state-of-the-art sub-

group discovery and rule learning algorithms by experimentally evaluating them on the UCI data sets, using the same data sets as in previous research of Stecher et al. (2014). All the competitors are used with their default parameters from their corresponding software platforms. In order to determine useful default configurations for our algorithms, we employ a data set hold-out methodology for parameter setting with the goal of finding the optimal configuration without tuning the algorithms to a particular data set.

The rest of this paper is organized as follows. Section 2 provides the necessary background on rule learning and subgroup discovery, followed by the introduction of the coverage space and an illustrative example, explaining the advantages of using inverted heuristics in rule refinement. It also summarizes the findings of Stecher et al. (2014) concerning the use of inverted heuristics in rule learning. Section 3 is concerned with subgroup discovery presenting the DoubleBeam-SD algorithm and its variants, followed by a description of the experimental setting and the obtained results. Section 4 outlines the DoubleBeam-RL algorithm for classification rule learning, followed by a description of the experimental setting, and the presentation of experimental results. Finally, Section 5 presents the conclusions and ideas for further work.

2. Rule learning: Background and related work

Rule learning is a standard symbolic data analysis technique used for constructing understandable models and patterns. Its main advantage over the other data analysis techniques is its simplicity and comprehensibility of its outputs. Rule learning has been extensively used both in predictive and descriptive rule learning settings, where by applying different rule evaluation heuristics different trade-offs between the consistency and coverage of constructed rules can be achieved.

This section first presents a short overview of classification rule learning and subgroup discovery. It introduces the coverage space used as a tool for studying the properties of different heuristics and presents the idea of using two separate heuristics for rule refinement and rule selection illustrated on a selected UCI data set. The section ends with the description of closely related work regarding the use of inverted heuristics in classification rule learning.

2.1. Classification rule learning

The task of classification rule learning is to find models which would ideally be *complete* (cover all positive examples, or at least most of the

positives), and *consistent* (not cover any negative examples, or at most a very small number of negatives). Multi-class classification problems can be dealt with by using the one-versus-all approach, which learns one rule set for each class, where the examples labeled with the chosen class are considered as positive target class examples, and all examples of other classes as negatives.

There are numerous classification rule learning algorithms, the most popular being AQ, CN2 and Ripper. The AQ algorithm (Michalski, 1969), which was the first to propose the covering algorithm for rule set construction, is a top-down beam search algorithm that uses a random positive example as a seed for finding the best rule. The CN2 algorithm (Clark and Niblett, 1989) combines the ideas from the AQ algorithm and the decision tree learning algorithm ID3 (Quinlan, 1983), given the similarity of rule learning to learning decision trees, where each path from the root of the tree to a tree leaf can be viewed as a separate rule. It constructs an ordered decision list by learning rules describing the majority class examples in the training set. Once the learned rule is added to the decision list, all the covered examples, both positive and negative, are removed from the training data set, and the rule induction process is continued on the updated training set. Ripper (Cohen, 1995) is the first rule learning algorithm that effectively overcomes the overfitting problem and is thus a very powerful rule learning system. The algorithm constructs rule sets for each of the class values. Initially, the training data set is divided into a *growing* and a *pruning* set. Rules are learned on the growing set, and then pruned on the *pruning* set by incrementally reducing the error rate on the pruning set. A pruned rule is added to the rule set if the description length of the newly constructed rule set is at most d bits longer (a parameter) than the already induced rule set. Otherwise, the rule learning process is stopped. Similarly to the CN2 algorithm, when a new rule is added to the rule set, all the instances covered by that rule are removed from the growing set. In addition to pruning the rules before adding them to induced rule set, Ripper prevents rules overfitting in a post-processing phase in which the learned rule set is optimized and the selected rules are re-learned in the context of the other rules. FURIA (Hühn and Hüllermeier, 2009) is a classification rule learning algorithm which extends the Ripper algorithm by learning fuzzy rules.

Despite its long history, rule learning is still actively researched and routinely applied in practice. For example, Napierala and Stefanowski (2015) use rule learning with argumentation to tackle imbalanced data sets, and Ruz (2016) explores the order of instances in seeding rules to improve the classification accuracy. Minnaert et al. (2015) discuss the importance of

proper rule evaluation measures for improving the accuracy of classification rule learning algorithms. They also introduce multi-criteria learning and investigate a Pareto front as a trade-off between comprehensibility and accuracy of rule learners.

In a line of research started by Parpinelli et al. (2002), rule learning is turned into an optimization problem using an ant colony optimization approach. The initial rule learning algorithm, named Ant-Miner, worked for nominal attributes only, but was later improved by Pičulin and Robnik-Šikonja (2014) to efficiently handle numeric attributes. Classification rule learning has been a vivid topic of research also in inductive logic programming and relational data mining. For example, Zeng et al. (2014) developed the QuickFOIL algorithm that improves over the original FOIL algorithm (Quinlan and Cameron-Jones, 1993).

Learning rules can be regarded as a search problem (Mitchell, 1982). Search problems are defined by the structure of the search space, a search strategy for searching through the search space, and a quality function (a *heuristic*) that evaluates the rules in order to determine whether a candidate rule is a solution or how close it is to being a solution to be added to the rule set, i.e. the final classification model. The search space of possible solutions is determined by the model language bias (Fürnkranz et al., 2012). In propositional rule learning, the search space consists of all the rules of the form $targetClass \leftarrow Conditions$, where $targetClass$ is one of the class labels, and $Conditions$ is a conjunction of features. Features have the form of $A_i = v_{ij}$ (attribute A_i has value v_{ij}).

For learning a single rule, most learners use one of the following search strategies: *general-to-specific* (*top-down hill-climbing*) or *specific-to-general* (*bottom-up*), where the former is more commonly used. Whenever a new rule is to be learned, the learning algorithm initializes it with the *universal rule* $\mathbf{r}^\top$. This is an empty rule that covers all the examples, both positive and negative. In the rule refinement phase, conditions are successively added to this rule, which decreases the number of examples that are covered by the rule. Candidate conditions are evaluated with the goal of increasing the consistency of the rule while maintaining its completeness, i.e. a good condition excludes many negative examples and maintains good coverage on the positive examples.

Heuristic functions are used in order to evaluate and compare different rules. Different heuristics implement different trade-offs between these two objectives. While CN2 and Ripper use entropy as the heuristic evaluation measure, numerous other heuristic functions have been proposed in rule learning—for a variety of heuristics and their properties the interested reader

is referred to (Fürnkranz et al., 2012). The most frequently used heuristics in rule learning are:

Precision:

$$\mathrm{h}_{prec}(p,n) = \frac{p}{p+n} \tag{1}$$

Laplace:

$$\mathrm{h}_{lap}(p,n) = \frac{p+1}{p+n+2} \tag{2}$$

m-estimate:

$$\mathrm{h}_{m\text{-}est}(p,n,m) = \frac{p + m \cdot \frac{P}{P+N}}{p+n+m} \tag{3}$$

where, for a given rule, arguments p and n denote the number of positive and negative examples covered by the rule (i.e. the true and false positives, respectively), and P and N in Equation (3) denote the total number of positive and negative examples in the data set. Given that these heuristics concern the problem of selecting the best of multiple refinements of the same base rule (the empty rule, universal rule), the values P and N can be regarded as constant, so that the above functions may be written as $h(p,n)$ depending only on the true and false positives.

Table 1 compares the DoubleBeam-RL classification rule learning algorithm (introduced in Section 4) to the state-of-the-art classification rule learners that were used in the experiments. CN2 and DoubleBeam-RL are beam search algorithms, while Ripper and SC-ILL are greedy algorithms, adding conditions to the rules which maximize their respective heuristics. The DoubleBeam-RL and SC-ILL algorithms use separate heuristics adapted for the refinement and selection phase of the rule learning process. Ripper is the only considered classification rule learning algorithm which employs rule pruning and optimization of rule sets in post-processing. The algorithms use different stopping criteria; for example, Ripper uses a heuristic based on minimum description length (MDL) principle.

2.2. Subgroup discovery

The goal of data analysis is not only building prediction models, but frequently the aim is to discover individual patterns that describe regularities in the data (Wrobel, 1997; Kralj Novak et al., 2005; Fürnkranz et al., 2012). This form of data analysis is used for data exploration and is referred to as *descriptive induction*. Subgroup discovery is a form of descriptive induction. The task of subgroup discovery is to find subgroups of examples which are

Algorithm	Type of search	Separate refinement heuristic	Stopping criterion	Rule pruning	Post-processing
CN2	beam	no	no beam improvement	no	no
Ripper	greedy	no	MDL	yes	yes
SC-ILL	greedy	yes	no negative examples covered	no	no
DoubleBeam-RL	beam	yes	*maxSteps*	no	no

Table 1: Comparison of the DoubleBeam-RL algorithm to the state-of-the-art classification rule learners CN2, Ripper and SC-ILL.

sufficiently large while having a significantly larger distribution of target class instances than the original target class distribution.

Like in classification rule learning, individual subgroup descriptions are represented as rules in the form $targetClass \leftarrow Conditions$, where the $targetClass$ is the target class representing the property of interest, and $Conditions$ is a conjunction of features that are characteristic for a selected group of individuals.

Subgroup discovery is a special case of the more general task of rule learning. Classification rule learners have been adapted to perform subgroup discovery with heuristic search techniques drawn from classification rule learning. These algorithm also apply constraints, which are appropriate for descriptive rule learning. The research in the field of subgroup discovery has developed in different directions. Exhaustive methods, which include EXPLORA (Klösgen, 1996), SD-MAP (Atzmüller and Puppe, 2006) and APRIORI-SD (Kavšek et al., 2003), guarantee the optimal solution given the optimization criterion. The APRIORI-SD algorithm draws its inspiration from the association rule learning algorithm APRIORI (Agrawal and Srikant, 1994), but restricts it to constructing rules that have only the target variable (the property of interest) in their head, with *weighted relative accuracy* (WRACC), defined in Equation (5), used as a measure of rule quality. In order to improve the inferential power of the subgroup describing rules, the APRIORI-SD algorithm uses a post-processing step to reduce the generated rules to a relatively small number of diverse rules. This reduction is performed using the weighted covering method proposed by Gamberger and Lavrač (2000). When a rule is added to the induced rule set, weights

of examples covered by the rule are decreased. This allows the method to prioritize rules which cover yet uncovered examples, thus promoting the coverage of diverse groups of examples.

While the APRIORI-SD algorithm adapts the process of association rule learning to the context of subgroup discovery, the SD subgroup discovery algorithm (Gamberger and Lavrač, 2002) performs heuristic beam search, where rule quality is estimated using the generalization quotient heuristic

$$\mathrm{h}_g(p, n, g) = \frac{p}{n + g}, \tag{4}$$

where p is the number of *true positives*, n is the *number of false positives*, and g is the *generalization parameter*. High-quality rules will cover many target class examples and a low number of non-target examples. The number of tolerated non-target examples covered by a rule is regulated by the generalization parameter. For small g, more specific rules are generated while for bigger values of g the algorithm constructs more general rules. The interpretation of the rules produced by the SD algorithm is improved using the above mentioned weighted covering method in post-processing (Gamberger and Lavrač, 2000).

CN2-SD (Lavrač et al., 2004) is a beam search algorithm, which adapts the CN2 (Clark and Niblett, 1989) classification rule learner to subgroup discovery. CN2-SD has introduced a weighted covering algorithm, where examples that have already been covered by one of the learned rules are not removed from the training data set, but instead their weights are decreased. The authors propose and compare different measures for rule evaluation. They argue that the most important measure for subgroup evaluation is *weighted relative accuracy* (WRACC), referred to as *unusualness*, defined as follows

$$\mathrm{WRACC}(p, n) = \frac{p + n}{P + N} \cdot \left(\frac{p}{p + n} - \frac{P}{P + N} \right) \tag{5}$$

This measure reflects both the rule significance and rule coverage, as subgroup discovery is interested in rules with significantly different class distribution than the prior class distribution that cover many instances. WRACC is the measure of choice in our experimental work on subgroup discovery for comparing the quality of the induced subgroup describing rules.

Subgroup discovery was used also in the context of semantic data mining. Adhikari et al. (2014) have explained mixture models by applying semantic subgroup discovery system Hedwig (Vavpetič et al., 2013) to structure the search space and to formulate generalized hypotheses by using concepts from the given domain ontologies.

Algorithm	Type of search	Separate refinement heuristic	Stopping criterion	Post-processing
APRIORI-SD	exhaustive	no	minSup, minConf	yes
SD	beam	no	no beam improvement	yes
CN2-SD	beam	no	no beam improvement	no
DoubleBeam-SD	two beams	yes	*maxSteps*	optional

Table 2: Some properties of subgroup discovery algorithms DoubleBeam-SD, APRIORI-SD, SD, and CN2-SD.

Table 2 compares the DoubleBeam-SD algorithm (introduced in Section 3) to the state-of-the-art subgroup discovery algorithms APRIORI-SD, CN2-SD, and SD, which were used in the experiments. The latter algorithms use only a single heuristic for rule evaluation, designed to optimize the selection of best rules. The DoubleBeam-SD algorithms can use pairs of different heuristics (see Section 2.4) which can be applied to estimate rule quality in both the *refinement* and *selection* phases of the rule learning process. The DoubleBeam-SD algorithm stops the learning process after a predetermined number of steps (*maxSteps*). The SD and CN2-SD algorithms stop when there are no improvements of rules in the beam, i.e. when newly induced rules have lower quality than the rules already included in the beam. APRIORI-SD uses minimal support and coverage as the stopping criteria.

2.3. Coverage space

Fürnkranz and Flach (2005) introduced the coverage space as a formal framework for analyzing and visualizing the behavior of rule learning heuristics. The *coverage space* (Fürnkranz and Flach, 2005; Fürnkranz et al., 2012), referred to as the *PN space* when initially introduced by Gamberger and Lavrač (2002), enables us to plot the number of covered positive examples (true positives p) over the number of covered negative examples (false positives n). This results in a rectangular plot with values $\{0,1,...,N\}$ (where N is the total number of negative examples) on the horizontal axis and $\{0,1,...,P\}$ (where P is the total number of positive examples) on the vertical axis. Figure 1 shows a coverage space visualization. The principle of coverage spaces can be used to plot individual rules, as well as entire theories or models composed of a rule set or a decision list.

There are four points of special interest in a coverage space:

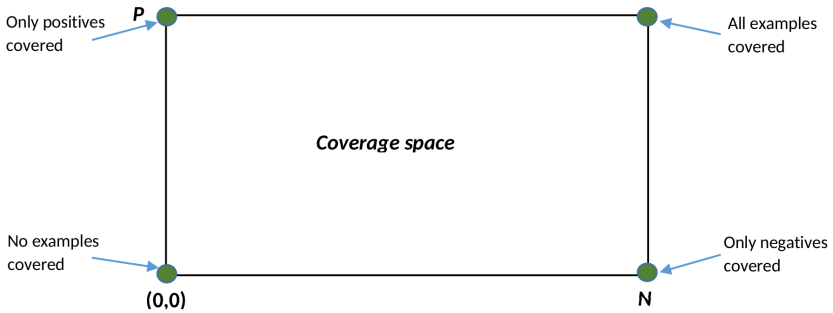

Figure 1: Visualization of coverage space with P (total of positives) and N (total of negatives).

- $(0, 0)$ marks the *empty theory*, denoted by $\mathbf{r}_\perp$. This theory covers no positive and no negative example.

- $(0, P)$ is the *perfect theory* which covers all positive and none of the negative examples.

- $(N, 0)$ is the *opposite theory*. It covers all negative, but no positive examples.

- (N, P) is the *universal theory*, denoted by $\mathbf{r}^\top$. This theory covers all the examples, regardless of their label.

The ultimate goal of learning is to reach the point of *perfect theory* in the coverage space, i.e. the point $(0, P)$. This will rarely be achieved in a single step. A set of rules will need to be constructed in order to achieve this objective. The purpose of heuristics used for rule evaluation is to determine how close a given rule is to this ideal point.

An isometric of a heuristic h is a line (or curve) in the coverage space that connects all points (p,n) for which $\mathrm{h}(p, n) = c$ for some constant value c. Several properties of heuristics can be seen from isometrics. As an example, Figure 2 shows the isometrics of precision, h_{prec}. These isometrics show that regarding precision all rules that cover only positive examples (points on the P-axis) achieve the best quality score, and all rules that cover only negative examples (points on the N-axis) achieve zero score. All other isometric values are obtained by rotation around the origin $(0, 0)$ for which the value of h_{prec} is undefined. Figure 2 presents also the disadvantage of precision, which is its inability to discriminate between rules with high and low coverage. For illustration, a rule that covers only one positive example

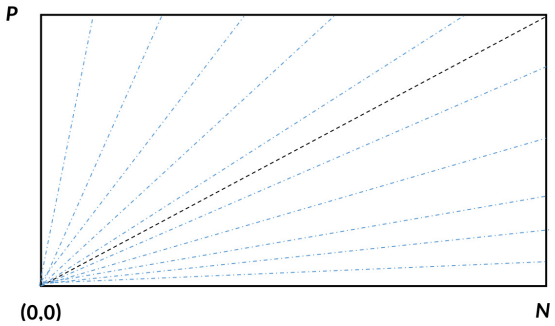

Figure 2: Isometrics for precision.

and no negative example will have better evaluation than a rule that covers a hundred positive examples and only one negative example.

The commonly used top-down strategy for rule refinement can be viewed as a path through the coverage space. Figure 3 illustrates rule refinement, where each point on the path corresponds to one further condition conjunctively added to the rule body. The path starts at the upper right corner, (N, P), with the universal rule $\mathbf{r}^{\top}$. By adding conditions to the rule, the number of covered positive and negative examples decreases and the path of the rule continues towards the origin (0, 0), which corresponds to the empty rule $\mathbf{r}_{\perp}$.

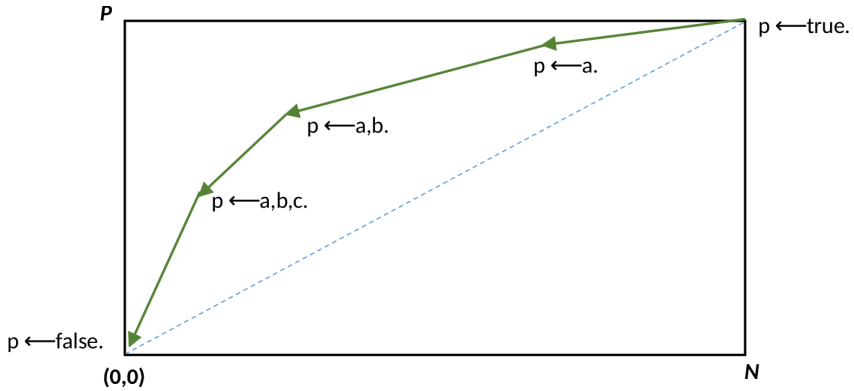

Figure 3: A path in the coverage space of a top-down specialization of a single rule. For simplicity, a comma is used to represent the conjunction operator.

2.4. Inverted heuristics

Rule learning algorithms rely on heuristic measures to determine the quality of the induced rules. Stecher et al. (2014) propose to distinguish

between rule refinement and rule selection heuristics in inductive rule learning. They argue that the nature of the separate-and-conquer rule learning algorithms opens up a possibility to use two different heuristics in the two fundamental steps of the rule learning process, i.e. rule refinement and rule selection. Using the coverage space they motivate separate evaluation of candidates for rule refinement and the selection of rules for the final theory. Stecher et al. (2014) further argue that the rule refinement step in a top-down search requires *inverted heuristics*, which can result in better rules. Such heuristics evaluate rules from the point of the current base rule, instead of the empty rule. In this way, while successively adding features to the rule (refinement), the learner favours rules with higher coverage of positive examples and thereby gives chance to rules with higher coverage to be finally selected with the selection heuristics.

Representations of the inverted heuristics in the coverage space reveal the following relationship with the basic heuristic:

$$\mathrm{u}(p, n) = \mathrm{h}(N - n, P - p) \tag{6}$$

where p and n denote the number of positive and negative examples covered by the rule, and P and N are not constant but depend on the predecessor of the currently constructed rule. For example, in the example illustrated in Figure 5, in the first step N and P correspond to the initial top-right corner (N, P) in the coverage space, but when refined to rule $p \leftarrow a$, the top-right corner is moved to point $\boldsymbol{B}$. The values of N and P will change respectively. Additionally, on the refinement path, N and P will be updated with the (N,P) coordinates of values of point $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{E}$, respectively in each next refinement iteration. Each of these points represent the base rule from which we observe the improvements of the consequent refinements.

Stecher et al. (2014) adapt the three standard heuristics for rule induction (introduced in Section 2.1): *precision*, *Laplace*, and *m-estimate*. The effect on these three heuristics is that the isometrics of their inverted variants do not rotate around the origin of the coverage space, but rotate around the point in the coverage space representing the base rule (the predecessor of the currently constructed rule). Consequently, the inverted heuristics have the following forms:

Inverted precision:

$$\mathrm{u}_{prec}(p, n) = \frac{N - n}{(P + N) - (p + n)}, \tag{7}$$

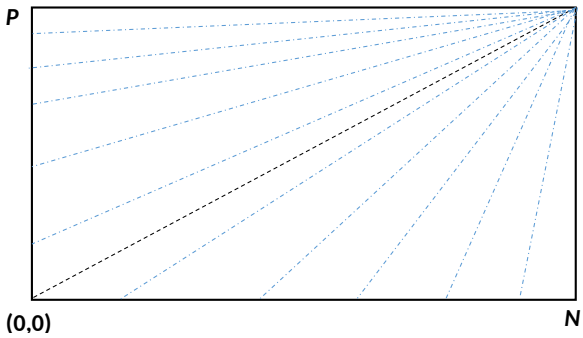

Figure 4: Isometrics of inverted precision.

Inverted Laplace:

$$\mathrm{u}_{lap}(p,n) = \frac{N-n+1}{(P+N)-(p+n-2)}, \tag{8}$$

Inverted m-estimate:

$$\mathrm{u}_{m\text{-}est}(p,n,m) = \frac{N-n+m\cdot\frac{P}{P+N}}{(P+N)-(p+n-m)}. \tag{9}$$

The inverted heuristics are not suited for rule selection. They do favor rules with high coverage but are also tolerant to covering negative examples. The isometrics of inverted precision in Figure 4 illustrate this propery.

For classification rule learning, Stecher et al. (2014) have shown that the combination of Laplace heuristic h_{lap} used in the rule selection step and the inverted Laplace heuristic u_{lap} used in the rule refinement step outperformed other combinations in terms of average classification accuracy. An interesting side conclusion from (Stecher et al., 2014) is that the usage of inverted heuristics in the rule refinement phase produces on average longer rules, which are claimed to be better for explanatory purposes.

We illustrate the advantage of using inverted heuristics in the refinement phase on the UCI (Lichman, 2013) *mushroom* data set. In Figure 5 we show the path in coverage space of top-down specialization of two rules for the class *poisonous* using different heuristics. Table 3 shows the descriptions of the coverage space points shown in Figure 5. The red path shows the top-down specialization of a rule using the h_{lap} heuristic. From all of the refinements of the universal rule, the refinement $odor = f$ has the steepest gradient from the origin $(0,0)$. Therefore, this rule is selected for further refinement. However, since the number of covered positive examples is $n = 0$,

the refinement process is terminated and the rule *odor* = *f* is also selected in the selection phase, covering 2160 positive and no negative examples.

The green path shows the top-down specialization of a rule using the q_{lap} heuristic. This heuristic prefers rules with high coverage of positive examples. It gives preference to rule refinements with the smallest angle between the line of the refinement and the horizontal axis, i.e. angles α, β, and γ in Figure 5. Top-down specialization continues until there are no covered negative examples or there are no possible refinements. In Figure 5 the refinement stops at point ***F***, where rule *veil-color* = *w*, *gill-spacing* = *c*, *bruises?* = *f*, *ring-number* = *o*, *stalk-surface-above-ring* = *k* is constructed, covering a total number of 2,192 positive examples and no negative examples. Using only a single selection heuristics this rule would be preferred to the rule depicted with the red path, but it is not achievable as a different choice was made already in the first step.

In summary, inverted heuristics prefer rules with high coverage of positive examples. The top-down specialization of a rule is steadily removing negative examples and some positive examples. This leaves the possibility that an additional refinement will construct a rule with the same or a higher number of covered positive examples than a rule constructed using a single heuristics which immediately maximizes its value.

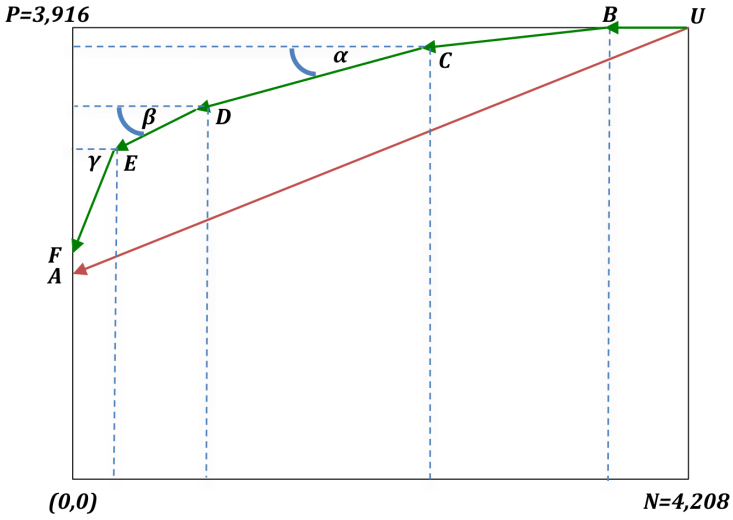

Figure 5: Comparison of rule refinement paths using standard heuristic and the inverted one. The red path shows rule constructed using h_{lap}. The green path shows rule refinement using q_{lap}.

Point	Rule			p	n
$\boldsymbol{U}$	p	←	true.	3,916	4,208
$\boldsymbol{A}$	p	←	odor = f.	2,160	0
$\boldsymbol{B}$	p	←	veil-color = w.	3,908	4,016
$\boldsymbol{C}$	p	←	veil-color = w, gill-spacing = c.	3,804	2,816
$\boldsymbol{D}$	p	←	veil-color = w, gill-spacing = c, bruises? = f.	3,188	160
$\boldsymbol{E}$	p	←	veil-color = w, gill-spacing = c, bruises? = f, ring-number = o.	3,152	144
$\boldsymbol{F}$	p	←	veil-color = w, gill-spacing = c, bruises? = f, ring-number = o, stalk-surface-above-ring = k.	2,192	0

Table 3: Description of coverage space points from Figure 3, illustrated on the mushroom data set, using target class p (*poisonous*).

2.5. Relation to previous work

Our work is closely related to previous work in rule learning and subgroup discovery. In particular, it explores the recommended approach by Stecher et al. (2014) for separation of rule refinement and rule selection and the use of different heuristics in the classification rule learning context. While rule induction algorithms and subgroup discovery algorithms typically use the same heuristic for rule refinement and rule selection, Stecher et al. (2014) argued that the nature of the separate-and-conquer algorithms offers the possibility of separating the two rule construction phases and their evaluation using two different heuristics.

In this paper we investigate the separation of the rule refinement and rule selection phase in both subgroup discovery and classification rule learning. Along with the phase separation we introduce two beams, each consisting of the best rules according to the refinement and selection heuristic, respectively.

Contrary to the approach of Stecher et al. (2014), where they compare the selection quality of the best rule refinement to the rule with the best selection quality, we compare the selection quality of all refined candidate rules to the selection quality of the best rules for selection (current selection beam members). In this way we expand the space of possible candidates for selection and increase the possibility of choosing a candidate with good selection quality, which might have been omitted in the refinement phase using the Stecher et al. (2014) approach. Additionally, our algorithm for rule learning builds rule sets for each target class of a given data set. This is different from the approach taken in (Stecher et al., 2014) where unordered decision lists are constructed.

This paper also significantly extends our previous work (Valmarska et al., 2015), where we reported on the initial findings regarding the use of inverted heuristics in subgroup discovery. In this paper, we introduce an additional heuristic, WRACC, which consequently proves to improve over other heuris-

tics in several settings. In addition, we propose a different approach to algorithm comparison, by first determining the default parameters for each algorithm and then comparing the algorithms on new data sets, using the default parameters. The establishment of default parameters is valuable for future users of the algorithms, as it offers a solid starting point for their use. In addition to the subgroup discovery algorithm, in this paper we also introduce a novel classification rule learning algorithm based on double beam and compare it to the state-of-the-art rule learning algorithms.

3. DoubleBeam Algorithm for Subgroup Discovery

The previously observed favorable properties of inverted heuristics in a classification setting provide a motivation to test the idea in the subgroup discovery context. For this purpose, we developed the DoubleBeam-SD subgroup discovery algorithm[1], which combines separate refinement and selection heuristics with the beam search. In the same fashion, we integrated the beam search and two separate heuristics in the classification rule learning setting, which we discuss in Section 4.

Contrary to conventional beam-search based algorithms such as CN2-SD (Lavrač et al., 2004), the DoubleBeam-SD algorithm for subgroup discovery maintains two separate beams, the *refinement beam* and the *selection beam*. Upon initialization, each beam is filled with the best single-condition rules according to their refinement and selection quality, respectively. The algorithm then enters a loop. In each iteration, rules of the form $targetClass \leftarrow Conditions$ from the refinement beam are refined by adding features to the $Conditions$ part of the existing rules. The resulting new rules are added to the refinement beam, which is ordered according to the refinement quality. Newly produced rules are then evaluated according to their selection heuristic and the selection beam is updated with the rules whose selection quality is better than the selection quality of the rules already stored in the beam. The algorithm exits the loop after the maximally allowed number of steps is reached. Another purpose of storing several rules in the selection beam is to allow post-processing where only the non-redundant subset of rules is retained (Gamberger and Lavrač, 2002). The DoubleBeam-SD algorithm is outlined in Algorithm 1.

In order to induce descriptions for subgroups of data instances which have not yet been covered by the previously constructed rules, we employ

[1]Code is available on github at `https://github.com/bib3rce/RL_SD`.

```
Input:      : E = P ∪ N
              E is the training set, |E| its size,
              tc is target class,
              P are positive examples (of class tc),
              N are negative examples (of classes ≠ tc).
Output:     : subgroup descriptions
Parameters: : minSupport,
              rbw is refinement beam width,
              sbw is selection beam width,
              rh is refinement heuristic,
              sh is selection heuristic
              maxSteps is maximal number of steps

CandidateList ← all feature values or intervals
for each candidate in CandidateList do
    evaluate candidate with rh
    evaluate candidate with sh
end
sort CandidateList according to the rh
for i = 0 to rbw do
    RB[i] ← CandidateList[i]
end
sort CandidateList according to the sh
for i = 0 to sbw do
    SB[i] ← CandidateList[i]
end
step ← 1
do
    refinedCandidates ← refine RB with CandidateList
    replace RB with refinedCandidates using rh
    updateSelectionBeam(SB, refinedCandidates, sh)
    step ← step + 1
while step ≤ maxSteps;
return SB
```

Algorithm 1: DoubleBeam-SD algorithm.

weighted covering, which reduces the weight of covered positive examples but does not remove them entirely. This required a modification of the method for updating the selection beam. Each time a positive example is covered by a rule that is already in the selection beam, the instance coverage count is increased and consequently the instance weight is decreased, which results in reducing the probability that the covered examples would be covered again by the rules constructed in the following iterations of the algorithm.

In this work, we used the harmonic and geometric weights for instance weighting. We also implemented removal of the already covered positive instances by assigning weight 0 to every instance already covered by some rule in the selection beam (method *zero weight*). Equations (10), (11) and

(12) show how the weight of a covered example is updated depending on the number of rules that cover it.

Geometric weight:

$$w_g(d_i) = \alpha^k, \tag{10}$$

where k is the number of rules that have already covered example d_i;

Harmonic weight:

$$w_h(d_i) = \frac{1}{k+1}, \tag{11}$$

where k is the number of rules that have already covered example d_i;

Zero weight:

$$w_z(d_i) = 0, \tag{12}$$

if example d_i is covered by at least one rule in the selection beam.

The weighted value of positive examples covered by a rule r (*weighted number of true positives*) is calculated using Equation (13).

$$wTP(r) = \sum_{i=1}^{|E|} w(d_i) \cdot c \quad \begin{cases} c = 1 & \text{if } r \text{ covers } d_i; \\ c = 0 & \text{otherwise.} \end{cases} \tag{13}$$

Note that zero weight can be understood as removing covered positive examples from the data set. This is not the same as no weighting, which means that instances are retained in the data set. As we use the selection beam, which keeps all the interesting subgroups, and the algorithm takes care that beam entries are not duplicated, no weighting might be sufficient. However, a practical reason to introduce instance weighting are possible redundancies in the attribute set. Without weighting we might get several different but redundant descriptions of the same instances in the beam, which unnecessary fill the beam and reduce the search space. The code of the **updateSelectionBeam** method is outlined in Algorithm 2.

Function **getBestRule** returns the rule with the best selection quality on the data set with updated weights. The selection quality of a rule is calculated according to the chosen selection heuristics. Function **updateWeights** updates the weights of the covered positive examples. The weights are updated according to the desired weight type i.e. geometric, harmonic or zero.

```
1  Method updateSelectionBeam(SB, refinedCandidates, sh)
       // current data
2      cData ← P ∪ N
       // candidates for selection
3      cs ← ∪SB
       // new selection beam
4      nSB ← {}
5      resetWeights(cData)
6      for i = 0 to sbw do
7          bestRule ← getBestRule(cs, cData, sh)
8          cs ← remove(cs, bestRule)
9          nSB{i} ← bestRule
10         cData ← updateWeights(cData, bestRule)
11     end
12     SB ← nSB
```

Algorithm 2: Method for updating the selection beam.

3.1. Experimental setting

For the purpose of algorithm evaluation, we use different combinations of refinement and selection heuristics, constituting the following DoubleBeam subgroup discovery variants:

SD-ILL (Inverted Laplace, Laplace), using $(\mathrm{u}_{lap}, \mathrm{h}_{lap})$ heuristics combination pair,

SD-IPP (Inverted Precision, Precision), using $(\mathrm{u}_{prec}, \mathrm{h}_{prec})$,

SD-IMM (Inverted M-estimate, M-estimate), using $(\mathrm{u}_{m\text{-}est}, \mathrm{h}_{m\text{-}est})$,

SD-IGG (Inverted Generalization quotient, Generalization quotient), using $(\mathrm{u}_g, \mathrm{h}_g)$,

SD-GG (Generalization quotient, Generalization quotient), using $(\mathrm{h}_g, \mathrm{h}_g)$, and

SD-WRACC (WRACC), using $(\mathrm{h}_{\mathrm{WRACC}}, \mathrm{h}_{\mathrm{WRACC}})$.

For the purpose of annotation, we prefix the variants of our DoubleBeam-SD with SD. The h_g heuristic is the generalization quotient proposed in (Gamberger and Lavrač, 2002) (Equation 4), while u_g is its inverted variant defined as $\mathrm{u}_g = \frac{N-n}{P-p+g}$. The weighted relative accuracy (WRACC) heuristic is defined in Equation (5). It was introduced in (Lavrač et al., 2004) to

Tuning data sets	C	E	A	F	**Evaluation data set**	C	E	A	F
breast-cancer	2	286	10	41	contact-lenses	3	24	5	9
car	4	1,728	7	21	futebol	2	14	5	27
glass	7	214	10	31	ionosphere	2	351	35	157
hepatitis	2	155	20	41	iris	3	150	5	14
horse-colic	2	368	23	72	labor	2	57	17	42
hypothyroid	2	3,163	26	60	mushroom	2	8,124	23	116
idh	3	29	5	14	primary-tumor	22	339	18	37
lymphography	4	148	19	52	soybean	19	683	36	99
monk3	2	122	7	17	tic-tac-toe	2	958	10	27
vote	2	435	17	32	zoo	7	101	18	134

Table 4: Number of classes (C), examples (E), attributes (A), and features (F) of the 20 data sets used in the experiments.

measure the unusualness of the induced subgroup describing rules. Note that WRACC is identical to its inverted variant (Stecher et al., 2014).

We compare three state-of-the-art subgroup discovery algorithms (SD, CN2-SD, and APRIORI-SD) and the proposed DoubleBeam-SD algorithm with six combinations of refinement and selection heuristics (SD-ILL, SD-IPP, SD-IMM, SD-IGG, SD-GG, and SD-WRACC). We test the DoubleBeam-SD algorithm with each of the six combinations of refinement and selection heuristics, both with and without using the weighted covering algorithm, and with and without using rule subset selection in the post-processing step described in (Gamberger and Lavrač, 2002). This resulted in 48 different combinations of the DoubleBeam-SD algorithm (6 refinement/selection combinations $\times$ 2 post-processing/no $\times$ 4 weighting/no).

We use SD, CN2-SD and APRIORI-SD implementations of algorithms that are available in the ClowdFlows platform (Kranjc et al., 2012). We use the same 20 UCI classification data sets as (Stecher et al., 2014) (see Table 4). In order to determine suitable settings, we randomly split the data sets into two groups: we use 10 randomly chosen data sets (shown on the left-hand side of Table 4) to determine default parameters of all competing methods, and the remaining 10 data sets (shown on the right-hand side of Table 4) to compare the best settings. The tuning of parameters is described in Section 3.2, while the methods comparison is presented in Section 3.3.

To compare the speed and scalability of the algorithms, we use the UCI *adult* data set which consists of 32,561 instances and 14 attributes. We do not use cross-validation on this data set but split it into training and test sets of different sizes.

Heuristics combination	Overall rank	Average rank	Post-processing	Weight type
WRACC	1	7.30	no	none
IMM	2	7.45	yes	none
GG	3	8.15	no	none
IGG	9	15.50	no	none
ILL	14	19.95	yes	none
IPP	21	24.45	no	zero

Table 5: Chosen variants of the DoubleBeam-SD algorithm. The overall rank is the rank of the algorithm among the 48 variants.

3.2. Default parameter setting

We use the 10 left-hand side data sets from Table 4 for setting default parameters of the algorithms. The SD algorithm and the APRIORI-SD algorithm are both trained using rule subset selection in the post-processing step, as described in (Gamberger and Lavrač, 2002). Originally, the CN2-SD algorithm does not use rule selection in the post-processing.

The algorithms are initially tested with 10-fold double-loop cross-validation on each of the 10 data sets used for parameter tuning (named *tuning data sets* in the rest of this paper). For each algorithm (both the newly proposed algorithms as well as the existing algorithms SD, CN2-SD and APRIORI-SD), a grid of possible parameter values is set in advance. The value of *minSup* is set to 0.01. Each training set of a given cross-validation iteration is additionally split into an internal training and testing subset. For each algorithm, models were built using the internal training subset and the parameters from its own parameter grid. Parameters maximizing the value of unusualness of the produced subgroups on the internal test subest are then chosen for building a model using the whole training set. In the evaluation, we use the subgroup discovery evaluation statistics proposed in (Kralj Novak et al., 2005) (originally implemented in the Orange data mining environment (Demšar et al., 2013)): *coverage*, *support*, *size*, *complexity*, *significance*, *unusualness* (WRACC), *classification accuracy*, and *AUC*.

We compute average ranks of the 48 combinations of the DoubleBeam-SD algorithm with respect to the unusualness (WRACC) of the produced subgroup describing rules. For each combination of refinement and selection heuristics of algorithms described in Section 3.1 we chose the algorithm setting that had the best average ranking. The chosen algorithm settings are shown in Table 5.

The default set of parameters for each algorithm consists of the parameters which were chosen in the 10-fold double-loop cross-validation testing phase. This default set of parameters is used for cross-validation testing of

Data sets	SD	CN2-SD	APRIORI-SD	SD-ILL	SD-IPP-w-z	SD-IMM	SD-WRACC	SD-GG	SD-IGG
contact-lenses	0.032	0.071	0.027	0.039	0.035	0.021	0.047	**0.081**	**0.081**
futebol	0.000	0.009	0.005	0.005	0.003	**0.015**	0.000	0.006	0.005
ionosphere	0.099	0.111	0.000	0.083	0.032	0.105	**0.133**	0.105	0.107
iris	0.090	**0.200**	0.142	0.159	0.167	0.146	0.175	0.148	0.148
labor	0.080	**0.102**	0.041	0.081	0.085	0.085	0.098	0.095	0.094
mushroom	0.088	0.163	0.000	0.133	0.029	0.134	**0.191**	0.146	0.131
primary-tumor	0.011	0.009	0.008	0.006	0.006	0.017	**0.019**	0.014	0.014
soybean	0.025	0.037	0.000	0.035		0.036	**0.043**	0.037	0.035
tic-tac-toe	0.022	0.021	0.029	0.024	0.029	0.024	**0.041**	0.028	0.029
zoo	0.037	0.097	0.000	0.094	0.065	0.096	**0.100**	0.099	0.094

Table 6: Ten-fold cross-validation WRACC results for subgroup discovery algorithms with default parameters. The best values for each data set are written in bold. We compare existing SD, CN2-SD and APRIORI-SD algorithms with the proposed DoubleBeam algorithms with different refinement and selection heuristics.

the subgroup discovery algorithms on the remaining 10 data sets.

3.3. Experimental results

The WRACC values obtained from the 10-fold cross-validation testing on the 10 evaluation data sets with selected default parameters are shown in Table 6. These values are averaged over all the classes for every particular data set.

The results of the Nemenyi test following the Friedman test for statistical significance of differences between average values of WRACC are shown in Figure 6. It is evident that SD-WRACC algorithm produces the most interesting subgroups, which are statistically more unusual than the ones produced by the two state-of-the-art algorithms, the SD algorithm and the APRIORI-SD algorithm. However, there are no statistically significant differences between the six chosen variants of the DoubleBeam-SD algorithm and the CN2-SD algorithm. The DoubleBeam-SD algorithm with the combination $(\mathrm{h}_g, \mathrm{h}_g)$ produces statistically more unusual subgroups than the ones produced by the APRIORI-SD algorithm. The rest of the variants of the DoubleBeam-SD algorithm do not produce subgroup describing rules which are statistically more interesting than the ones produced by any of the tested algorithms.

Experimental results reveal that algorithms which use WRACC as their heuristic (the SD-WRACC algorithm and the CN2-SD algorithm) produce rules which describe more interesting subgroups. The underperformance of the other considered variants of the DoubleBeam-SD algorithm is due to their respective heuristics, which are specialized towards finding prediction rules and not unusual rules.

The results of the Nemenyi test following Friedman test for statistical significance of differences of the average rule sizes are shown in Figure 7.

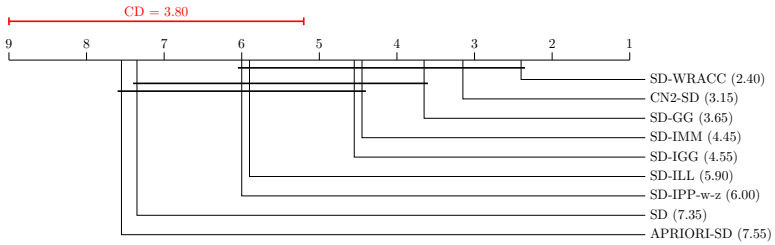

Figure 6: Nemenyi test on ranking of subgroup discovery algorithms regarding average WRACC values with a significance level of 0.05.

The DoubleBeam-SD algorithm with the combination $(\mathrm{h}_g, \mathrm{h}_g)$ produces subgroups which are on average described by the longest rules. The SD-GG algorithm generates subgroups described by rules that are statistically longer only than the ones produced by the SD algorithm and the SD-IPP algorithm with zero-weight covering. There is no statistical evidence that the SD-GG algorithm produces longer rules than other evaluated algorithms. Consequently, these results do not confirm that the DoubleBeam-SD algorithm with inverted refinement heuristic produces statistically longer subgroup descriptions than all other subgroup discovery algorithms. This is slightly surprising taking into account the findings of (Stecher et al., 2014) in the classification rule learning setting.

Table 7 presents the performance of subgroup discovery algorithms on the *adult* data set in terms of their WRACC score. We split the data set in the 70:30 ratio, leading to 22,793 training and 9,768 testing instances. The SD-WRACC algorithm produced the most interesting rules, followed by the CN2-SD algorithm. The results are in accordance with the results presented in Figure 6. The algorithms SD-ILL, SD, APRIORI-SD and SD-GG gen-

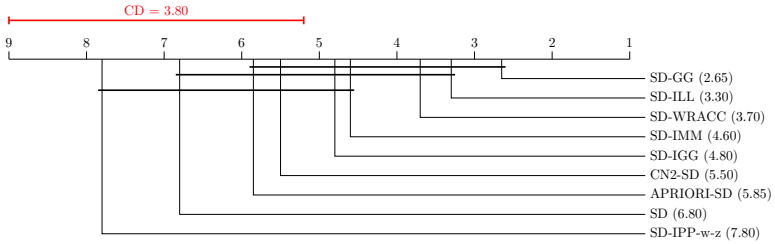

Figure 7: Nemenyi test on ranking of average rule sizes for subgroup discovery algorithms in the second experimental setting with a significance level of 0.05. Note that algorithms are ordered according to the average length of generated rules—rank 1 would indicate the algorithm producing the longest rules.

Measure	SD	CN2-SD	APRIORI-SD	SD-ILL	SD-IPP-w-z	SD-IMM	SD-WRACC	SD-GG	SD-IGG
WRACC	0.023	0.043	0.041	0.011	0.012	0.028	0.076	0.025	0.024
ARL	2.800	2.150	2.700	2.800	1.300	2.100	2.100	2.600	2.500

Table 7: Performance comparison of subgroup discovery algorithms using WRACC score and average rule length (ARL) on the UCI *adult* data set. The data set is split in 70:30 ratio. Rules are induced using the default parameters.

erated the longest rules; the SD-ILL and SD-GG algorithms produced the longest rules also on data sets from Figure 7.

Figure 8 presents the *training* times of subgroup discovery algorithms with different numbers of training instances from the *adult* data set. The APRIORI-SD algorithm is the slowest, followed by the SD-IPP-w-z and CN2-SD algorithms. The other subgroup discovery algorithms are comparable in terms of training time and allow for processing of relatively large data sets.

To the users of subgroup discovery algorithms we recommend the use of the SD-WRACC algorithm with the selection beam width set to 5, and no example weighting or post-processing. Results show that this algorithm on average outperforms other subgroup discovery algorithms considered in this work.

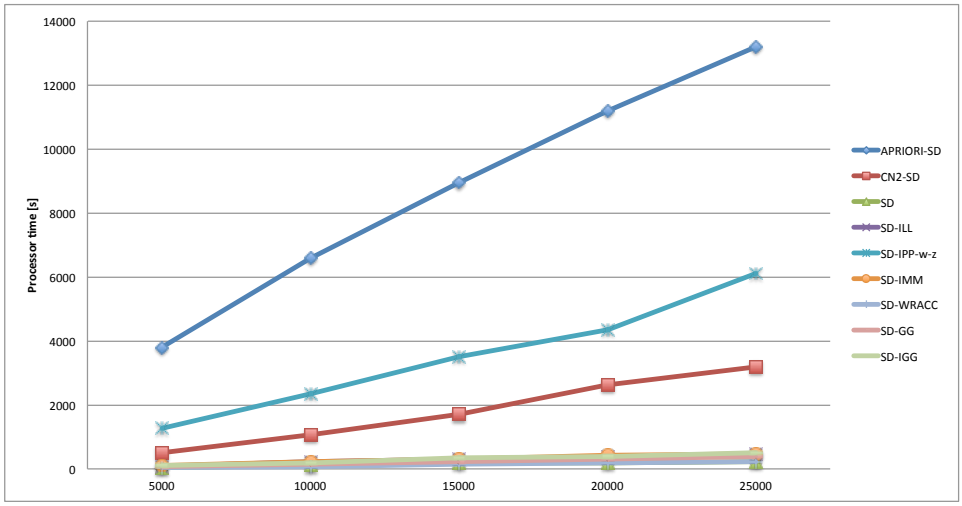

Figure 8: Comparison of training times for subgroup discovery on the *adult* data set. The horizontal axis shows the number of training instances and the vertical axis shows the training time in seconds.

4. DoubleBeam Algorithm in Classification Rule Learning

The idea of using two separate heuristics for rule refinement and selection as well as using inverted heuristics in refinement phase was proposed and successfully tested by Stecher et al. (2014). The previous section shows that this idea can also be successful in subgroup discovery, where we tested it using a double beam search approach. As Stecher et al. (2014) do not use beam search in rule learning, an obvious extension is to use double beam also in classification rule learning.

In order to test the influence of different selection heuristics, refinement heuristics, selection beam width, and refinement beam width, we implemented a DoubleBeam classification rule learning (DoubleBeam-RL) algorithm. This algorithm is adaptation of the DoubleBeam-SD algorithm. It uses a combination of refinement and selection heuristics for each phase of rule learning. The algorithm has two beams, the selection beam and the refinement beam, where during the process of generating rules it holds potential candidates for refinement and selection, based on their selection and refinement quality. For learning a decision list, it employs the commonly used separate-and-conquer strategy (Fürnkranz, 1999): each time a rule is generated for a given target class, the positive examples covered by the rule are removed from the data set. The algorithm continues to learn new rules for the same target class on the updated data set as long as rules with a minimal acceptable quality are induced, i.e. if the rule covers more positive than negative examples and covers more positive examples than a chosen threshold (in our case a threshold of 2). The final result is a rule set with acceptable rules for the given target class.

Basically, for learning a single rule, a single beam (in the refinement phase) is sufficient, unless we want to produce a collection of rules which are post-processed later. If not, we shall set the selection beam width to 1, as we do in our experiments. The DoubleBeam-RL algorithm is outlined in Algorithm 3. The function for generating a single rule when a data set, selection heuristics, refinement heuristics, selection beam width, and refinement beam width are given is outlined in Algorithm 4.

4.1. Experimental setting

We perform experimental evaluation in two steps. In the first step we determine default parameters for the five best combinations of refinement and selection heuristics on the same randomly chosen 10 data sets in the left-hand side of Table 4. In the second step, we use 10 fresh data sets (the right-hand side of Table 4) to compare these five best configurations with

Input: : $E = P \cup N$
E is the training set, $|E|$ its size,
tc is target class,
P are positive examples (of class tc),
N are negative examples (of classes $\neq tc$).
Output: : R, (R is rule set for tc)
Parameters: : rh is refinement heuristic,
sh is selection heuristic,
rbw is refinement beam width,
sbw is selection beam width.

```
// rule set for target class tc
R ← {}
// current data
cData ← E
do
    rule ← generateRule(cData, tc, rh, sh, rbw, sbw)
    R ← R + rule
    cData ← removePositiveCovered(cData, rule, tc)
while not satisfied;
return R
```

Algorithm 3: DoubleBeam-RL algorithm.

```
Function generateRule (dataset, tc, rh, sh, rbw, sbw)
    // candidates for best rule
    bRC ← DoubleBeam-SD(dataset, tc, rh, sh, rbw, sbw)
    bestRule ← getBestRule(bRC)
    return bestRule
```

Algorithm 4: Function for generating rules using two heuristics.

two state-of-the-art algorithms for rule learning, Ripper (Cohen, 1995) and CN2 (Clark and Niblett, 1989). We use the Weka (Hall et al., 2009) implementation of Ripper and the Orange (Demšar et al., 2013) implementation of the CN2 algorithm. For both algorithms we use the default parameters set by their software platforms, respectively. For comparison, we also include the results from the best performing algorithm from Stecher's (Stecher et al., 2014) experimental work, named SC-ILL.

The quality of the induced rules is measured in terms of the classification accuracy (CA). The process of parameter tuning and variant selection is described in Section 4.3. We also report the average rule length of produced rules.

2,192	p	←	veil-color = w, gill-spacing = c, bruises? = f, ring-number = o, stalk-surface-above-ring = k.
864	p	←	veil-color = w, gill-spacing = c, gill-size = n, population = v, stalk-shape = t.
336	p	←	stalk-color-below-ring = w, ring-type = p, stalk-color-above-ring = w, ring-number = o, cap-surface = s, stalk-root = b, gill-spacing = c.
264	p	←	stalk-surface-below-ring = s, stalk-surface-above-ring = s, ring-type = p, stalk-shape = e, veil-color = w, gill-size = n, bruises? = t.
144	p	←	stalk-shape = e, stalk-root = b, stalk-color-below-ring = w, ring-number = o.
72	p	←	stalk-shape = e, gill-spacing = c, veil-color = w, gill-size = b, spore-print-color = r.
44	p	←	stalk-surface-below-ring = y, stalk-root = c.

Table 8: Decision list learned for class p (*poisonous*) in the mushroom data set using Stecher's approach with refinement heuristic u_{lap} and selection heuristic h_{lap}. The number of positive examples covered by each rule is also shown. No rule covers any of the negative examples.

4.2. Illustrative example

We compare our approach with the approach of Stecher et al. (2014) with an illustrative example. For the purpose of this comparison, we chose the same set of attributes used in the mentioned work. Rules in both decision lists are generated with u_{lap} as the refinement heuristic and h_{lap} as the selection heuristic. The width of both refinement and selection beam is set to 1. Figure 8 shows the decision list learned for the class *poisonous* on the data set *mushroom* using the algorithm presented in (Stecher et al., 2014), whereas Figure 9 presents the rule set learned by our DoubleBeam rule learning algorithm.

Results from Tables 8 and 9 suggest that our approach tends towards finding even more complete rules than the approach taken by Stecher et al. (2014). The algorithm produces on average shorter rules which include more or the same number of examples. The DoubleBeam-RL algorithm is able to detect features that do not contribute to the overall improvement of the rules. Such example is the `bruises? = f` feature. In the first rule from Stecher's decision rule, the 2,192 covered examples are covered by the conjunction of the following features: `veil-color = w, gill-spacing = c, ring-number = o, stalk-surface-above-ring = k`. Feature `bruises? = f` was selected during the refinement phase, but does not contribute anything to the final result.

This difference is due to the nature of the applied algorithms. Stecher's approach is to refine a rule using the inverted heuristics until there are only positive examples covered and then returns the best rule on the refinement path. This approach leads to eliminating possible refinements of a certain rule due to their lower refinement quality, even though their selection quality

is very high; in our case, one of the possible refinements has even better selection quality than the final rule, chosen by the Stecher's approach. The DoubleBeam-RL algorithm on the other hand, considers the selection quality of the refined candidates and the rules already in the selection beam. It simultaneously checks for rules with best refinement and selection quality and keeps track of all the best rules found in the refinement process.

As an example, consider rules from Table 3. After the universal rule is refined, the best candidate for further refinement in both approaches is `p ← veil-color = w.` The DoubleBeam-RL algorithm saves this rule as a candidate for refinement, but chooses rule `p ← odor = f` as its candidate for best rule. In the next iteration, once more the two algorithms have the best candidate for refinement, `p ← veil-color = w, gill-spacing = c.` There is no change in the selection beam of the DoubleBeam-RL algorithm, where the selection quality of `p ← odor = f` (1.000) is better than the selection quality of `p ← veil-color = w, gill-spacing = c` (0.575). In the third step, best rule for refinement is `p ← veil-color = w, gill-spacing = c, bruises? = f.` Both algorithms will continue with the refinement of this rule, however, the selection beam of the DoubleBeam-RL algorithm will be updated with a refinement of `p ← veil-color = w, gill-spacing = c,` leading to rule `p ← veil-color = w, gill-spacing = c, stalk-surface-above-ring = k,` whose selection quality is the same as the selection quality of the rule already stored in the beam (1.000). When the DoubleBeam-RL algorithm is faced with choosing between two rules with the same selection quality, it always chooses the rule that has covered more positive examples. In case the decision is not straight-forward, it chooses the shortest among the rules in question. The top-down specialization will continue for both algorithms. The algorithm proposed by Stecher will stop when there are only positive examples covered or there is no possible further refinement. At the end, the algorithm will return the rule with the best selection quality among all the rules on the refinement path. As it is evident from our example, this will result with longer rules which can have lower coverage than the rules selected by the DoubleBeam-RL algorithm. The DoubleBeam-RL algorithm stops after a predefined number of steps, and returns the rule with the best selection quality among all the investigated refinements.

4.3. Default parameter setting

In the experiments performed to determine the default parameter values for the DoubleBeam-RL algorithm, we use all combinations of the following heuristics in refinement and selection phase:

2,228	p	←	p ← gill-spacing = c, veil-color = w, stalk-surface-above-ring = k.
864	p	←	gill-color = b .
336	p	←	stalk-color-above-ring = w, gill-spacing = c, stalk-root = b, stalk-color-below-ring = w, gill-attachment = f, cap-surface = s, ring-number = o, ring-type = p.
264	p	←	stalk-shape = e, bruises? = t, gill-size = n, gill-attachment = f, stalk-surface-above-ring = s, stalk-surface-below-ring = s, ring-type = p.
144	p	←	stalk-shape = e, bruises? = f, stalk-root = b, stalk-color-below-ring = w, gill-attachment = f.
72	p	←	stalk-shape = e, spore-print-color = r.
8	p	←	veil-color = y.

Table 9: Rule set learned for the class p (*poisonous*) in the mushroom data set using DoubleBeam rule learning algorithm with refinement heuristic $\mathfrak{u}_{lap}$, selection heuristic h_{lap}, and both refinement and selection beam width set to 1. The number of positive examples covered by each rule is shown on the left. No rule covers any negative examples.

- Laplace h_{lap} - L,
- Inverted Laplace $\mathfrak{u}_{lap}$ - IL,
- Precision h_{prec} - P,
- Inverted Precision $\mathfrak{u}_{prec}$ - IP,
- M-estimate $\mathrm{h}_{m\text{-}est}$ - M,
- Inverted -M-estimate $\mathfrak{u}_{m\text{-}est}$ - IM, and
- Weighted Relative ACCuracy $\mathrm{h}_{\mathrm{WRACC}}$ - W.

As an example, the abbreviation RL-ILL indicates that $\mathfrak{u}_{lap}$ was used as a refinement heuristic, and h_{lap} as a selection heuristic in the DoubleBeam-RL algorithm. This resulted in 49 variants of the DoubleBeam-RL algorithm. Each variant is tested on the same 10 randomly chosen data sets that were used for parameter tuning in the subgroup discovery context.

The value of the selection beam width is fixed to 1 in all variants (see Section 4). In order to select the default width of the refinement beam for each variant of the DoubleBeam-RL algorithm, we perform a 10-fold double-loop cross-validation of each variant on each of the tuning data sets from Table 4. Each tuning data set is divided into training and test set. Each training data set is additionally split into internal training and test subset. A separate model is induced on the internal training subset for each of the possible parameter values. These models are then evaluated using the internal test subset. The parameter values that maximize the value of classification accuracy are chosen as parameters for the construction of the model using the initial training data set. The final cross-validation value

of classification accuracy (CA) for each fold is calculated using the model induced with the chosen best parameters and the corresponding test data set.

For each heuristic combination we collected the best parameters for refinement beam width across the tested 10 data sets. The most frequently selected parameter was chosen as a default parameter for the considered combination. Our experiments showed that for each variant of the rule learning algorithm, the most accurate rules are induced when we use refinement beam width with value 1. This means that the selected best parameters make our algorithm identical to the rule learning algorithm proposed in (Stecher et al., 2014), with the exception that our algorithms can select the best rule from each refinement step (line 2 of Algorithm 4 and line 18 of Algorithm 1), while in (Stecher et al., 2014) only the final refined rule is selected. This seemingly small difference leads our algorithm to form shorter rules with better coverage and affects also the classification accuracy as presented in Section 4.4.

Out of the 49 variants of the DoubleBeam-RL algorithm, we eventually selected the following five variants, which had the best average rank performance on the 10 tuning data sets: RL-MM ($\mathrm{h}_{m\text{-}est}$,$\mathrm{h}_{m\text{-}est}$), RL-ILM ($\mathrm{q}_{lap}$,$\mathrm{h}_{m\text{-}est}$), RL-WM ($\mathrm{h}_{\mathrm{WRACC}}$,$\mathrm{h}_{m\text{-}est}$), RL-IPM ($\mathrm{q}_{prec}$,$\mathrm{h}_{m\text{-}est}$), and RL-PM ($\mathrm{h}_{prec}$,$\mathrm{h}_{m\text{-}est}$).

4.4. Experimental results

We compare the selected best rule learning algorithm (using the five chosen variants of rule selection heuristics) with two state-of-the art algorithms, Ripper and CN2, and the best performing algorithm from Stecher et al. (2014)'s work, named SC-ILL. The classification accuracy (CA) values obtained from the 10-fold cross-validation testing on the 10 evaluation data sets from the right-hand side of Table 4 with default parameters are shown in Table 10.

The Friedman test for statistical differences in CA showed that there are no significant differences between the algorithms which is confirmed by the confidence intervals of the Nemenyi test in Figure 9. Nevertheless, the approach taken by Stecher et al. (2014) yields the best results (average rank is 3.00). Three of our five chosen variants of the DoubleBeam-RL algorithm have a better average rank than the Ripper algorithm. The chosen variants of our algorithm for rule learning on average perform better than the CN2 algorithm.

An interesting observation is that among all the algorithms with two heuristics, the one with the least search performs best i.e. the Stecher

Data sets	RL-MM	RL-ILM	RL-WM	RL-IPM	RL-PM	Ripper	SC-ILL	CN2
contact-lenses	0.750	0.750	0.750	0.750	0.750	0.750	**0.875**	0.683
futebol	0.700	0.700	0.700	0.700	0.700	0.571	0.571	**0.800**
ionosphere	0.900	0.861	0.858	0.875	0.914	0.897	**0.932**	0.906
iris	0.920	0.920	0.920	0.920	0.920	**0.953**	**0.953**	0.893
labor	0.773	0.820	0.720	0.820	**0.827**	0.771	0.825	0.720
mushroom	0.999	**1.000**	**1.000**	**1.000**	0.997	**1.000**	**1.000**	**1.000**
primary-tumor	0.401	0.407	**0.410**	0.395	0.345	0.392	0.360	0.345
soybean	0.921	0.908	0.903	0.909	0.852	0.915	**0.924**	0.883
tic-tac-toe	**0.982**	0.976	0.892	0.974	0.980	0.978	0.976	0.818
zoo	0.872	0.892	0.882	0.892	0.823	0.871	0.921	**0.961**

Table 10: Ten-fold cross-validation CA results for rule learning with default parameters. Best values are written in bold.

et al. (2014) approach. The explanation for this could be the *over-searching* phenomenon (Quinlan and Cameron-Jones, 1995; Janssen and Fürnkranz, 2009), which indicates that the amount of search shall be adjusted specifically to a data set and search heuristics employed.

The results of the Friedman test and post-hoc Nemenyi test for statistical significance of differences between average rule length of rules induced by the chosen variants of the DoubleBeam-RL algorithm and the state-of-the-art algorithms for classification rule learning are shown in Figure 10. The results suggest that the variant that uses the inverted heuristic in refinement phase, RL-IPM, induces rules which are statistically longer than the rules induced by the standard refinement heuristic, RL-PM. The results in Figure 10 are in accordance with the conclusions drawn by Stecher et al. (2014). The approach taken by Stecher et al. (2014), SC-ILL, produces longest rules, while the CN2 algorithm produces rules with the shortest average rule length. These rules are significantly shorer than the rules produced by the SC-ILL, the Rl-IPM, and the RL-ILM algorithm. Note the the average rule length is calculated as the ratio between the sum of all conditions across all induced

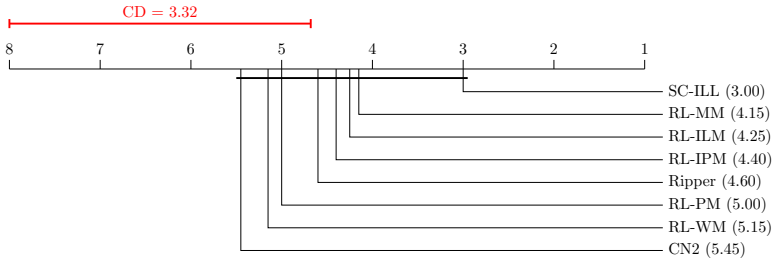

Figure 9: Nemenyi test on ranking of classification accuracy values with a significance level of 0.05.

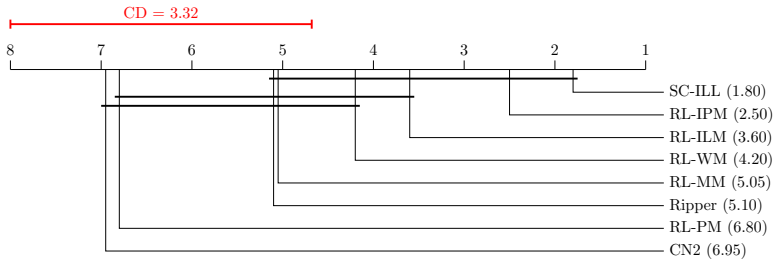

Figure 10: Nemenyi test on ranking of average classification rule length with a significance level of 0.05.

rules and the total number of rules in the model.

Table 11 shows the performance comparison of classification rule learning algorithms on the *adult* data set in terms of classification accuracy and average rule length. Results reveal that all versions of the DoubleBeam-RL algorithm produce rules with better classification accuracy than the CN2 algorithm. Three DoubleBeam-RL algorithms (RL-ILM, RL-WM, and RL-IPM) slightly outperform the Ripper algorithm. Comparison of the results obtained with the algorithms RL-IPM and RL-PM confirm the conclusions of Stecher et al. (2014): when an inverted heuristic is used in the refinement phase, the produced rules tend to be longer and have better classification accuracy.

Figure 11 presents the *training* times of classification rule learning algorithms with different numbers of training instances from the *adult* data set. The times can only give a rough picture of the algorithms' performance, as the algorithms are not implemented on the same platform: we use the Ripper implementation from Weka, CN2 from Orange, the other algorithms are implemented in Python. The training times of the SC-ILL algorithm are not included due to excessive time consumption of the algorithm. Figure 11 shows that the Ripper algorithm is the fastest classification rule learner. Algorithms RL-IPM and RL-ILM have almost identical training times and are the most inefficient. An interesting observation is that DoubleBeam-RL

Measure	RL-MM	RL-ILM	RL-WM	RL-IPM	RL-PM	Ripper	SC-ILL	CN2
CA	0.834	0.851	0.852	0.854	0.835	0.845	/	0.815
ARL	2.909	2.824	1.938	2.684	1.214	4.333	/	2.531

Table 11: Comparison of classification accuracy (CA) and average rule length (ARL) of classification rule learning algorithms on the UCI *adult* data set. Data set is split in 70:30 ratio. Models are induced using estimated default parameters. SC-ILL results are not included as its training took more than 5 hours of CPU time.

algorithms, which use inverted heuristics in their refinement phase, produce slightly more accurate models (Table 11) than their Laplace counterparts at the cost of being less efficient. Figure 11 reveals that algorithms RL-ILM, RL-IPM, RL-MM, and RL-WM may use less time in spite of larger training set. Further investigation revealed relatively large variance of measured times. For specific points the mentioned algorithms produce models with fewer rules and fewer conditions.

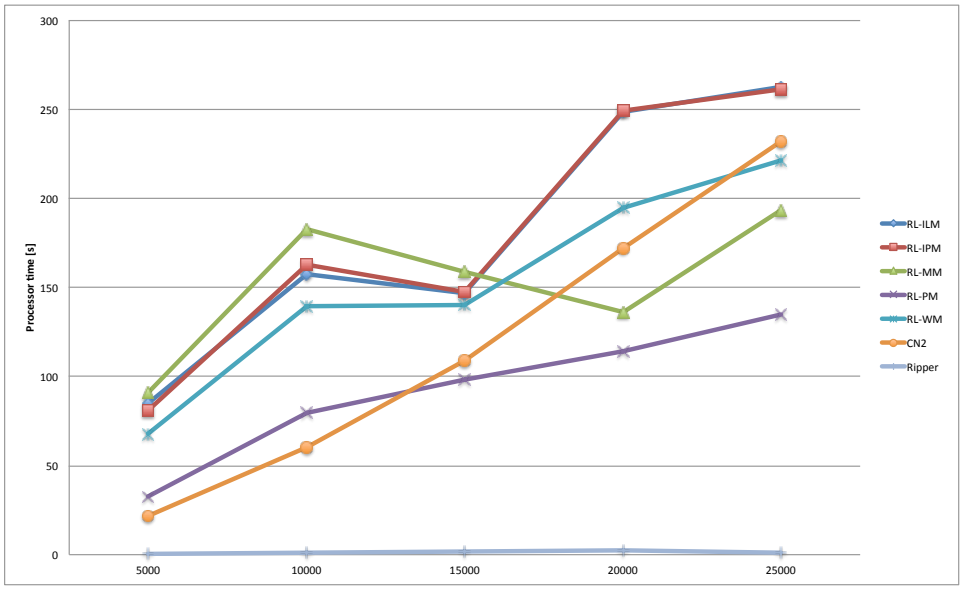

Figure 11: Comparison of training times for classification rule learning algorithms on the *adult* data set. The horizontal axis shows the number of training instances and the vertical axis shows the training time in seconds.

Based on our experimental work, there can be no clear recommendation for the user which algorithm to use, as the differences in classification accuracy are not statistically significant. However, several algorithms with two heuristics produces on average more accurate rules than Ripper and CN2, the most accurate being SC-ILL and RL-MM. For large data sets where computational efficiency is crucial, Ripper is clearly the best choice.

5. Conclusions

This paper introduces two new algorithms for rule learning, one for subgroup discovery and one for classification rule learning. Both algorithms

use beam search and offer the possibility to use separate heuristics for rule refinement and rule selection.

The experiments were performed on 20 UCI data sets. The performance of each of the considered algorithms depends on its parameters. In order to systematically choose the default parameters for each algorithm, we initially performed 10-fold double-loop cross-validation training on 10 randomly chosen data sets. The conclusions about the performance of the discussed algorithms are obtained after ten-fold cross-validation testing on the remaining 10 data sets, which have not been used for parameter tuning and are exclusively used for algorithm evaluation.

The experiments indicate that the subgroup describing rules created using the SD-WRACC algorithm are more interesting than the subgroups induced by other state-of-the-art subgroup discovery algorithms. The difference between most of the algorithms are not statistically significant, however SD-WRACC and CN2-SD produce statistically significantly more interesting rules than SD and APRIORI-SD.

In the context of classification rule learning we proposed a new, the DoubleBeam-RL algorithm, which offers the possibility for using separate rule refinement and selection heuristics. Among the tested 49 variants of refinement and selection heuristics inside the DoubleBeam-RL algorithm and their comparison with Ripper and CN2, the best performing variants in terms of classification accuracy were the algorithms that use the m-estimate as their selection heuristic. In particular, the best performing variant of the DoubleBeam-RL algorithm was the variant that uses the m-estimate both as its selection and refinement heuristic. The differences are, however, not statistically significant. The algorithms which use inverted heuristic perform slightly better than the algorithms using the standard heuristics (RL-IPM and RL-ILM compared to RL-PM and RL-WM). All five of our algorithms perform better than the CN2 algorithm, and three of our algorithms perform better than the Ripper algorithm.

The main advantage of DoubleBeam-SD and DoubleBeam-RL algorithms is their ability to use separate heuristics for the refinement and selection phase of rule learning. Different heuristics can take advantage of the data properties and contribute to better rules (rules with improved unusualness or rules with improved accuracy). The experimental results suggest that both algorithms provide rules with comparable or better quality than those obtained by the state-of-the-art algorithms for rule learning and subgroup discovery, respectively. The use of two beams in combination with separate heuristics for each phase of the learning processes widens the algorithms' search space thus improving the probability of finding better quality rules.

However, this also increases the chances of data overfitting, an aspect which our algorithms do not explicitly address at this point.

In contrast to the APRIORI-SD algorithm which uses exhaustive search, the DoubleBeam-SD algorithm is a heuristic search algorithm (similar to the SD and the CN2-SD algorithm). Despite being faster than the APRIORI-SD algorithm and ability to handle medium size data sets, the current DoubleBeam-SD algorithm is still not able to handle large data sets, due to space and time complexity. In fact, this is one of the main disadvantages of all rule learning algorithms using a covering approach. Lower memory consumption could be achieved with more efficient data structures, while significant speedups could be gained with instance sampling and feature subset selection, as well as with parallelization of the algorithms. Due to two beams, large degree of parallelization could be achieved with DoubleBeam algorithms.

While our DoubleBeam-SD and DoubleBeam-RL algorithms show promising results, their increased search power demands further research in terms of stopping criteria and rule pruning heuristics. Using a post-processing rule pruning step similar to the Ripper is a promising research direction. We plan to explore also the rule pruning method proposed by Sikora (2011).

Experimental results on subgroup discovery revealed the advantage of using WRACC over the traditional rule learning heuristics in obtaining interesting subgroups. We believe that developing new heuristics specialized for the detection of interesting subgroups is a promising research path.

Subgroup discovery is a useful approach in the analysis of medical data. In line with our work on Parkinson's disease data (Valmarska et al., 2016), we plan a case-study comparing results of different subgroup discovery algorithms on Parkinson's disease patients data set. In order to increase the interpretability of the induced subgroup describing rules we also plan on presenting a method for subgroup visualization. In this way we will assist experts (e.g. physicians) in their decision whether a certain subgroup discovery rule is interesting and relevant for their work.

Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (research core fundings No. P2-0209 and P2-0103). This research has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No. 720270 (HBP SGA1). We would like to thank Julius Stecher for interesting discussions on inverted heuristics.

References

Adhikari, P. R., Vavpetič, A., Kralj, J., Lavrač, N., and Hollmén, J. (2014). Explaining Mixture Models through Semantic Pattern Mining and Banded Matrix Visualization. In *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, pages 1–12.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499.

Atzmueller, M. (2015). Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49.

Atzmüller, M. and Puppe, F. (2006). SD-map - A fast algorithm for exhaustive subgroup discovery. In *Proceedings of Knowledge Discovery in Databases, PKDD 2006*, pages 6–17.

Bay, S. D. and Pazzani, M. J. (2001). Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246.

Bibal, A. and Frénay, B. (2016). Interpretability of machine learning models and representations: an introduction. In *Computational Intelligence and Machine Learning, Proceedings of European Symposium on Artificial Neural Networks ESANN 2016*, pages 77–82.

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

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123.

Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., and Zupan, B. (2013). Orange: Data mining toolbox in Python. *Journal of Machine Learning Research*, 14:2349–2353.

Dong, G. and Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999*, pages 43–52.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

Fürnkranz, J. and Flach, P. A. (2005). ROC 'n' rule learning - Towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77.

Fürnkranz, J., Gamberger, D., and Lavrač, N. (2012). *Foundations of Rule Learning*. Springer.

Gamberger, D. and Lavrač, N. (2002). Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527.

Gamberger, D. and Lavrač, N. (2000). Confirmation Rule Sets. In *Proceedings of Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000*, pages 34–43.

Grosskreutz, H. and Rüping, S. (2009). On subgroup discovery in numerical domains. *Data Mining and Knowledge Discovery*, 19(2):210–226.

Hall, M. A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.

Hühn, J. and Hüllermeier, E. (2009). Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319.

Janssen, F. and Fürnkranz, J. (2009). A re-evaluation of the over-searching phenomenon in inductive rule learning. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 329–340. SIAM.

Kavšek, B., Lavrač, N., and Jovanoski, V. (2003). APRIORI-SD: adapting association rule learning to subgroup discovery. In *Advances in Intelligent Data Analysis V, 5th International Symposium on Intelligent Data Analysis, IDA*, pages 230–241.

Klösgen, W. (1996). Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271.

Kralj Novak, P., Lavrač, N., and Webb, G. I. (2009). Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403.

Kralj Novak, P., Lavrač, N., Zupan, B., and Gamberger, D. (2005). Experimental comparison of three subgroup discovery algorithms: Analysing brain ischemia data. In *Proceedings of the 8th International Multiconference Information Society*, pages 220–223.

Kranjc, J., Podpečan, V., and Lavrač, N. (2012). ClowdFlows: A cloud based scientific workflow platform. In *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012*, pages 816–819.

Lavrač, N., Kavšek, B., Flach, P. A., and Todorovski, L. (2004). Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188.

Lavrač, N., Železný, F., and Flach, P. A. (2002). RSD: Relational subgroup discovery through first-order feature construction. In *Proceedings of the 12th International Conference on Inductive Logic Programming*, pages 149–165.

Lichman, M. (2013). UCI machine learning repository.

Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*.

Minnaert, B., Martens, D., De Backer, M., and Baesens, B. (2015). To tune or not to tune: rule evaluation for metaheuristic-based sequential covering algorithms. *Data Mining and Knowledge Discovery*, 29(1):237–272.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18(2):203–226.

Napierala, K. and Stefanowski, J. (2015). Addressing imbalanced data with argument based rule learning. *Expert Systems with Applications*, 42(24):9468 – 9481.

Parpinelli, R. S., Lopes, H. S., and Freitas, A. A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on evolutionary computation*, 6(4):321–332.

Piculin, M. and Robnik-Šikonja, M. (2014). Handling numeric attributes with ant colony based classifier for medical decision making. *Expert systems with applications*, 41(16):7524–7535.

Quinlan, J. and Cameron-Jones, R. (1995). Oversearching and layered search in empirical learning. In *Proceedings of the 14th international joint conference on Artificial intelligence, IJCAI'95*, volume 2, pages 1019–1024. Morgan Kaufmann Publishers Inc.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. *Machine learning. An artificial inteligence approach.*

Quinlan, J. R. and Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of Machine Learning: European Conference on Machine Learning - ECML 1993*, pages 3–20.

Ruz, G. A. (2016). Improving the performance of inductive learning classifiers through the presentation order of the training patterns. *Expert Systems with Applications*, 58:1 – 9.

Sikora, M. (2011). Induction and pruning of classification rules for prediction of microseismic hazards in coal mines. *Expert Systems with Applications*, 38(6):6748–6758.

Stecher, J., Janssen, F., and Fürnkranz, J. (2014). Separating rule refinement and rule selection heuristics in inductive rule learning. In *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014*, pages 114–129.

Valmarska, A., Miljkovic, D., Robnik-Šikonja, M., and Lavrač, N. (2016). Multi-view Approach to Parkinson's Disease Quality of Life Data Analysis. In *Springer, Lecture Notes in Computer Science*.

Valmarska, A., Robnik-Šikonja, M., and Lavrač, N. (2015). Inverted heuristics in subgroup discovery. In *Proceedings of the 18th International Multiconference Information Society*.

Vavpetič, A., Novak, P. K., Grčar, M., Mozetič, I., and Lavrač, N. (2013). Semantic Data Mining of Financial News Articles. In *Proceedings of the 16th International Conference Discovery Science, DS 2013*, pages 294–307.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery, PKDD 1997*, pages 78–87.

Zeng, Q., Patel, J. M., and Page, D. (2014). Quickfoil: Scalable inductive logic programming. *Proceedings of Very Large Databases Conference*, 8(3):197–208.