Separating Rule Refinement and Rule Selection Heuristics in Inductive Rule Learning

Julius Stecher, Frederik Janssen, and Johannes Fürnkranz

Technische Universität Darmstadt, Knowledge Engineering
jlstecher@gmail.com, {janssen,juffi}@ke.tu-darmstadt.de

Abstract. Conventional rule learning algorithms use a single heuristic for evaluating both, rule refinements and rule selection. In this paper, we argue that these two phases should be separated. Moreover, whereas rule selection proceeds in a bottom-up specific-to-general direction, rule refinement typically operates top-down. Hence, in this paper we propose that criteria for evaluating rule refinements should reflect this by operating in an inverted coverage space. We motivate this choice by examples, and show that a suitably adapted rule learning algorithm outperforms its original counter-part on a large set of benchmark problems.

1 Introduction

Separate-and-conquer or *covering* rule learning algorithms [6,8] proceed by first learning a single rule (*conquer*) followed by the removal of all examples that are covered by this rule (*separate*). The remaining examples are then used to learn the next rule (return to the conquer step). For learning a rule, most algorithms use a *top-down hill-climbing* search that starts with the *universal rule* covering *all* examples, and subsequently add conditions that optimize a *heuristic*. Typical heuristics trade off *consistency* and *coverage*, i.e., they prefer rules that cover as few *negative* and as many positive examples as possible [7,9].

Typically, such a heuristic is used in two different places in this process: (i) for judging *rule refinements*, i.e., to select which of the refinements of the current rule will be further explored, and (ii) for *rule selection*, i.e., to finally decide which of the refinements that have been explored is added to the rule set. In this paper, we argue that these tasks should be treated separately, i.e., evaluated with separate heuristics. Moreover, we argue that the rule refinement step in a top-down search requires *inverted heuristics*, which evaluate rules from the point of view of the current base rule instead of the empty rule. We will motivate this with an example, show the derivation of such *inverted heuristics* in coverage space, and demonstrate empirically that they lead to improved performance.

We start with a brief recapitulation of separate-and-conquer rule learning, heuristics and coverage spaces (Section 2). In Section 3, we then motivate why rule refinement and rule selection should be separated, and show how the commonly used heuristics precision, Laplace, and m-estimate can be inverted to better reflect a top-down search for refinements. We will also see that other heuristics, such as weighted relative accuracy, are invariant to such inversions. Finally, in Section 4, we evaluate the use of inverted heuristics for evaluating rule refinements experimentally on 20 UCI datasets.

Algorithm 1. Procedure Separate-And-Conquer

```
Data: TrainingData
Result: theory $\mathcal{R}$

1 Start with empty theory $\mathcal{R}$
2 while positive examples left in TrainingData do
3     Rule r = findBestRule(TrainingData)
4     if positiveCovered(r) $\leq$ negativeCovered(r) then
5         break
6     $\mathcal{R} = \mathcal{R} \cup \mathbf{r}$
7     remove all covered examples from TrainingData
8 return $\mathcal{R}$
```

2 Separate-and-Conquer Rule Learning

In this section, we briefly recapitulate the necessary foundations for our contribution, the separate-and-conquer rule learning algorithm (Section 2.1), coverage spaces (Section 2.2), and rule learning heuristics (Section 2.3).

2.1 Algorithm

Most rule learning algorithms follow a so-called separate-and-conquer or covering strategy to learn from P positive and N negative training examples. This algorithm proceeds by learning one rule at a time, while removing all examples that are covered by each rule from the dataset. This is repeated until no examples remain, i.e., until all examples are covered, or until the best found rule covers more negative than positive examples. Algorithm 8 shows the basic algorithm, as it has also been described in [6,8].

In contrast to algorithms producing an unordered rule set, we consider the learned rule set as a *decision list* made up of an ordered list of rules. A decision list ends with a *default rule*, which unconditionally applies the majority class label to any example that is not covered by one of the previous rules in the list. At classification time, the ordered rule list is checked from top to bottom, assigning to each example the class label of the head of the first rule that matches the example.

For finding individual rules, we focus on the most commonly used *top-down hill-climbing* strategy, which is shown in Algorithm 11. Whenever it needs to learn a new rule the algorithm initializes it with the *universal rule* $\mathbf{r}^{\top}$, which covers *all* examples. By adding conditions to this rule, the amount of covered examples will decrease with each iteration, thereby increasing the consistency of the rule by focusing on removing more negative examples than positive examples. How much consistency is gained depends on the particular condition that is selected as a refinement of the rule in each iteration. This choice depends on a *heuristic* function h, which is applied to all possible rule refinements, choosing the refinement that scores best after applying the heuristic to all refinements. It is easy to see that the importance of a good heuristic is vital for learning a theory w.r.t. consistency and coverage as it is the only type of guidance the rule learner can make use of during the training process.

Algorithm 2. Procedure findBestRule

```
Data: TrainingData
Result: best rule $\mathbf{r}_{best}$
1  $\mathbf{r}_{best} = \emptyset$
2  bestValue = heuristic($\mathbf{r}_{best}$)
3  repeat
4      get possible refinements
5      forall the refinements ref do
6          evaluation = heuristic(ref)
7      $\mathbf{r}_{ref}$ = best refined rule
8      if heuristic($\mathbf{r}_{ref}$) $\geq$ bestValue then
9          $\mathbf{r}_{best} = \mathbf{r}_{ref}$
10 until no refinements left;
11 return $\mathbf{r}_{best}$
```

2.2 Coverage Space

Coverage spaces have been introduced as a formal framework for analyzing and visualizing the behavior of rule learning heuristics [7]. A coverage space plots the number of covered positive examples (the *true positives* p) over the number of covered negative examples (the *false positives* n), resulting in a rectangular plot with the values $\{0, 1, ..., N\}$ on the horizontal axis and $\{0, 1, ..., P\}$ on the vertical axis. This principle can then be used to both plot entire theories consisting of an ordered rule list (the decision list) as well as individual rules.

The following points of the coverage space are of special interest (cf. also Figure 1):

- $(0, 0)$ is the *empty theory*. It does not cover any examples, neither positive nor negative ones. A *bottom-up* learning algorithm would start at this point and successively add rules.
- $(0, P)$ is the *perfect theory* covering all positive, but no negative examples.
- $(N, 0)$ is the *opposite theory* covering all negative, but no positive examples.
- (N, P) is the *universal theory*. It covers all examples regardless of their label.

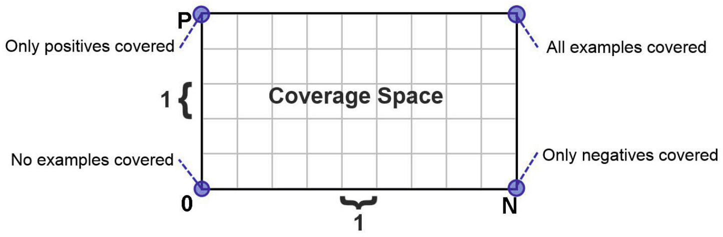
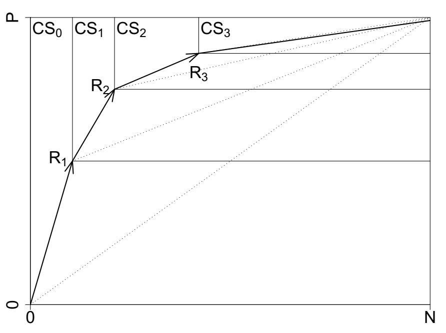


Fig. 1. Coverage space visualization with P total positive examples and N total negative examples



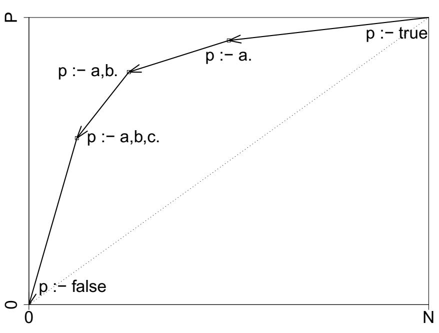


Fig. 2. Paths in coverage space for (*left*) the covering strategy of learning a rule set by adding one rule at a time and (*right*) top-down specialization of a single rule

Fürnkranz and Flach [7] have shown that learning a rule set one rule at a time may be viewed as a path through coverage space, where each point on the path corresponds to the addition of a rule to the theory. Figure 2 shows the coverage path for a theory with three rules. Each point $\mathcal{R}_i = \bigcup_{j=1}^{i}\{\mathbf{r}_j\}$ represents the rule set consisting of the first i rules. Adding a rule moves the induced rule set to the next point $\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{\mathbf{r}_{i+1}\}$.

Removing the covered (positive and negative) examples has the effect of switching to a subspace of the original coverage space, using the last learned rule as the new origin. Thus the path may also be viewed as a sequence of nested coverage spaces CS_i. Each new rule is evaluated relative to the origin $(0, 0)$ of this new coverage space. For example, precision would pick the rule with the steepest ascent from the origin.

The commonly used top-down strategy for rule refinement, on the other hand, successively specializes a rule by adding the most promising condition to the rule body. Just as with adding rules to a rule set, successive rule refinements describe a path through coverage space (Figure 2, right). However, in this case, the path starts at the upper right corner with the universal rule $\mathbf{r}^{\top}$, and successively proceeds towards the origin, which corresponds to the empty rule $\mathbf{r}_{\perp}$.

2.3 Rule Learning Heuristics

Any rule learning algorithm relies on some sort of measure to determine the quality of a rule; this is done with the help of a *heuristic function* h. Most heuristics implement a trade-off between consistency and coverage favoring rules that cover as many positive examples as possible (optimizing coverage) while keeping the amount of negative examples covered small (optimizing consistency). Thus, the computed value depends mostly on p (positive examples covered) and n (negative examples covered). Since for some of the examined heuristics (e.g. the m-estimate as well as the modifications suggested later) the values of P (total positive examples) and N (total negative examples) must be known, for most purposes a heuristic can be defined as a function

$$h : (p, n, P, N) \rightarrow \mathbb{R}$$

For the problem of selecting the best of multiple refinements of the same base rule, the values P and N can be regarded as constant, so that the function may be written as $\mathrm{h}(p, n)$ depending only on the true and false positives.[1] Such a formulation also allows to visualize the behavior of these heuristics by plotting their *isometrics* in coverage space [7]. Isometrics are lines in coverage space that connect points (n, p) that share the same heuristic value $\mathrm{h}(p, n)$. Figure 3 shows examples of such isometric plots for two heuristics discussed below.

For the experiments in this paper, we will focus on three common base heuristics with slightly different but related properties:

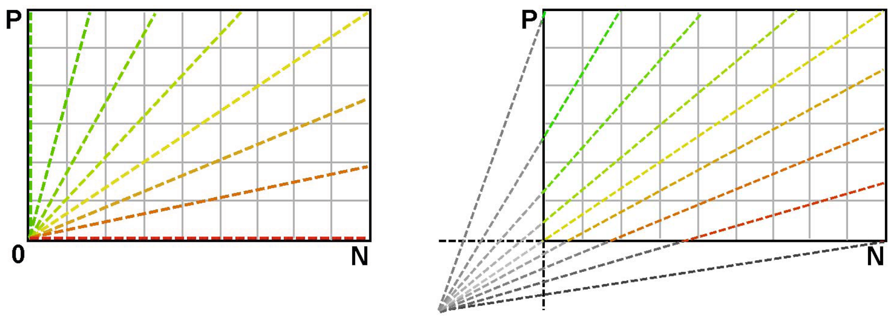
Precision: $\mathrm{h}_{prec}(p, n) = \frac{p}{p+n}$
Precision prefers a rule $\mathbf{r}_1$ to another rule $\mathbf{r}_2$ if $\mathbf{r}_1$ covers a larger *percentage* of positive examples. Note that this does not take into account coverage – a rule covering one positive and no negative examples will score the highest possible value, while a rule covering all positive and one negative example will score slightly lower. Thus a theory learned with the help of the precision heuristic is likely to overfit the training data with a bad performance when generalizing to new or noisy data. This can also be seen from its visualization in coverage space (Figure 3(a)), which shows that the isometrics of the precision heuristic rotate around the origin $(0, 0)$, and that therefore all points on the P axis receive the same evaluation.

Laplace: $\mathrm{h}_{lap}(p, n) = \frac{p+1}{p+n+2}$
The Laplace heuristic reduces some of the overfitting drawbacks (bad generalization) of precision while following the same general intent of maximizing (mostly) consistency. Starting the p and n counts at 1 instead of 0, the origin of the isometrics shifts to $(-1, -1)$. The effects of this change is that rules on the P-axis not sharing the same value anymore. For example, if two rules $\mathbf{r}_1$ and $\mathbf{r}_2$ cover no negative examples, but $\mathbf{r}_1$ covers 2 positives while $\mathbf{r}_2$ only covers 1, the resulting heuristic values are $\mathrm{h}_{lap}(\mathbf{r}_1) = 0.75$ and $\mathrm{h}_{lap}(\mathbf{r}_2) = 0.66$, whereas evaluating both rules with precision would have yielded $\mathrm{h}_{prec}(\mathbf{r}_1) = \mathrm{h}_{prec}(\mathbf{r}_2) = 1.0$.

m-Estimate: $\mathrm{h}_{mest}(p, n) = \frac{p+m\cdot\frac{P}{P+N}}{p+n+m}$
The m-estimate may be considered as a generalization of the Laplace heuristic. It follows the same idea, but features a parameter m that allows to shift the origin of the rotation, which is fixed at $(0, 0)$ for precision and at $(-1, -1)$ for Laplace to any place along the negative extension of the diagonal of the coverage space. Essentially, this has the effect of initializing all coverage counts with m examples, which are distributed according the overall distribution. For the special case $m = 0$, the m-estimate equals precision, and for $m \to \infty$, it approximates *weighted relative accuracy* (WRA), which means that its isometrics approach parallel lines with a slope of $\frac{P}{P+N}$ (the a priori distribution).[2] For the algorithm that we use in our experiments, an optimal value of

[1] Some heuristics also include additional parameters such as the length of the rule. However, this is often implicitly captured (longer rules correlate with lower coverage), and adding them does not necessarily yield increased performance [9].

[2] WRA is defined as $\frac{p+n}{P+N} \cdot (\frac{p}{p+n} - \frac{P}{P+N})$, which is equivalent to $\frac{p}{P} - \frac{n}{N}$. We will not further consider it in this paper, for reasons that will be explained in Section 3.2.



(a) precision (b) m-estimate

Fig. 3. Visualization of the isometrics of precision and the m-estimate

$m = 22.466$ has been determined experimentally [9], and we will be use this value in our experiments as well. Figure 3(b) shows the isometrics for the m-estimate heuristic. It can be clearly seen that the isometrics rotate around a point in the negative space, which has the effect that points on the P-axis no longer receive the same evaluation.

3 Optimization via Modified Heuristics for Rule Refinement

In the standard separate-and-conquer implementation, we use the same heuristic function each time we want to evaluate an entire rule or a refinement of a rule to determine the current best rule and the best refinement w.r.t. the goals of the heuristic (usually coverage and consistency). The approach highlighted in this paper modifies this standard algorithm to use different heuristics for *rule selection* and *rule refinement*. In particular, we will propose to separate these two phases and show how to adapt the three heuristics mentioned above for top-down rule refinement.

3.1 Motivation

As we have seen in Section 2.2, top-down hill-climbing takes a path through coverage space, starting from the universal rule in its upper-right corner. Common rule learning algorithms evaluate each of the rules encountered on this path with a heuristic in the same coverage space. For example, precision would evaluate two candidate rules according to the steepest ascent from the origin, as it would do with rule selection. However, we argue that this evaluation is, in a way, irrelevant because, while it selects the best complete rule that can currently be added to the rule set, it does *not* select the best candidate for further refinement.

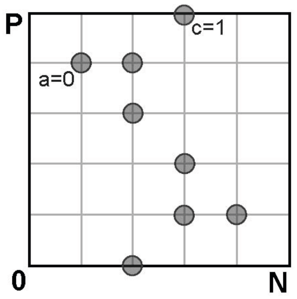
This illustrated in an example dataset with four binary attributes and a binary class attribute shown in Figure 4(a). The corresponding coverage statistics of all possible refinements are listed in Figure 4(b) and plotted in coverage space in Figure 4(c). According to precision h_{prec}, the refinement $a = 0$ is clearly the best choice, as is illustrated in Figure 4(d), whereas the refinement $c = 1$ would only be the third choice. However, we

$a\ b\ c\ d$	class	$a\ b\ c\ d$	class
0 1 1 1	+	0 1 1 0	+
0 1 1 1	+	0 0 1 1	+
0 0 1 0	−	1 1 1 0	−
1 1 1 0	−	1 0 1 1	+
1 0 0 1	−	1 0 0 1	−

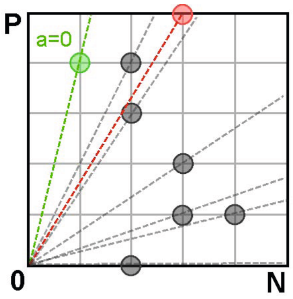
(a) Example dataset

condition	$p\ n$	condition	$p\ n$
$\mathbf{a = 0}$	**4 1**	$a = 1$	1 4
$b = 0$	2 3	$b = 1$	3 2
$c = 0$	0 2	$\mathbf{c = 1}$	**5 3**
$d = 0$	1 3	$d = 1$	4 2

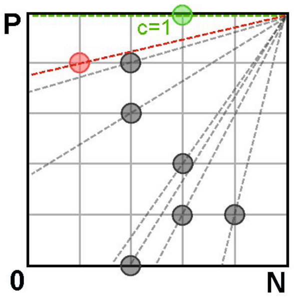
(b) Possible refinements



(c) Possible refinements in coverage space



(d) Precision selects $a = 0$ as best refinement



(e) $c = 1$ is a better choice from a top-down view

condition	$p\ n$	condition	$p\ n$
$a = 0 \wedge b = 0$	1 1	$a = 0 \wedge \mathbf{b = 1}$	**3 0**
$a = 0 \wedge c = 0$	0 0	$a = 0 \wedge c = 1$	4 1
$a = 0 \wedge d = 0$	1 1	$a = 0 \wedge \mathbf{d = 1}$	3 0

(f) refinements for $a = 0$

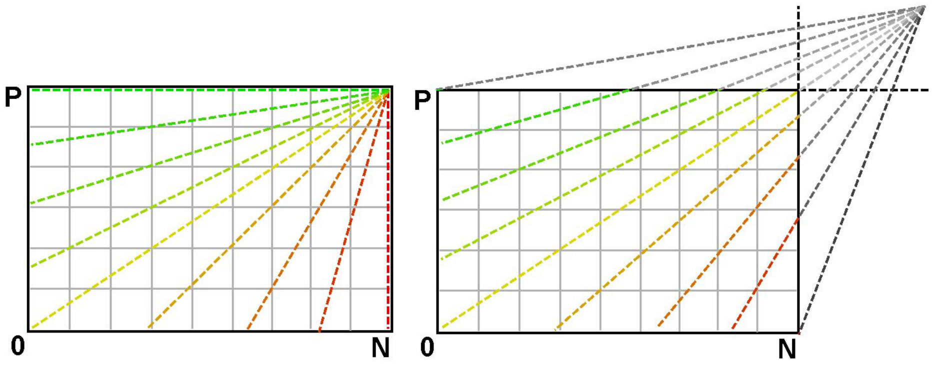
condition	$p\ n$	condition	$p\ n$
$c = 1 \wedge a = 0$	4 1	$c = 1 \wedge a = 1$	1 3
$c = 1 \wedge b = 0$	3 2	$c = 1 \wedge b = 1$	2 2
$c = 1 \wedge d = 0$	1 4	$c = 1 \wedge \mathbf{d = 1}$	**4 0**

(g) refinements for $c = 1$

Fig. 4. Example dataset with refinements

argue that $c = 1$ is a better choice for a refinement, because it covers more positive and negative examples and can thus be still refined into a rule that may be better than the first refinement. As the refinement $a = 0$ already has lost one positive example, further refinements will never cover 5 positive examples as is theoretically still possible when $c = 1$ is chosen. However, this choice can be obtained if we use a precision-like heuristic, whose isometrics do not rotate around the origin, but rotate around the base rule, as sown in the right part of Figure 4(e). Indeed, as can be seen from the further possible refinements of these two rules shown in Figures 4(f) and 4(g), the best refinement from the choice $a = 0$ is a rule that covers 3 positive and no negative examples (both $b = 1$ and $d = 1$ can be selected in this case). On the other hand, the precision-like heuristic whose isometrics rotate around the best rule, would end up in the final rule $c = 1 \wedge d = 1$, which covers 4 positive and no negative examples. This rule is preferable to the previous ones but could not be found with the conventional application of precision.

In the next section, we will derive top-down versions of heuristics that correspond to precision, Laplace, and the m-estimate.



(a) inverted precision (b) inverted m-estimate

Fig. 5. Visualization of the isometrics of the top-down versions of precision and the m-estimate

3.2 Adaptation of Heuristics to Top-Down Rule Refinement

Note that the three base heuristics (h_{prec}, h_{lap} and h_{mest}) all share similar isometrics, with the only difference being the origin (in the latter case, the location of the origin can be configured via the parameter *m*). As motivated in the previous section, we want to preserve this attribute, but shift the origin to the top right corner of the coverage space. The intention of this is that in our case the *rule refiner* follows the top-down strategy (starting off with the most general rule and successively adding conditions). We have to take into account that the values of P and N are not constant this time w.r.t. the heuristic function, but depend on the predecessor of the rule. This is because for our approach to work, we will want the origin of the isometrics to be placed at the point in coverage space corresponding to the base rule we want to refine, which will produce *nested* coverage spaces, and subsequently evaluate the refinements within the base rule's nested coverage space.

Figure 5 illustrates the intended behavior for the cases of precision and m-estimate. Instead of a rotation around the origin as in their original versions depicted in Figure 3, we aim for a rotation around the base rule, which is located in (N, P). Moreover, we also have to swap the positive and negative axes: While the best refinements starting from the origin lie on the P-axis, the best refinements starting from (N, P) lie on the N-axis of the coverage space. More precisely, we have to modify the heuristic in a way so that it holds that

$$\mathrm{\text{ɥ}}(p, n) = \mathrm{h}(N - n, P - p) \tag{1}$$

where ɥ is the *inverted* or *top-down heuristic* in the coverage space with dimensions P and N, whereas h is the original heuristic, but in a coverage space with swapped dimensions N and P.

For the three heuristics discussed in Section 2.3, it is straight-forward to see that we obtain the following expressions:

- ***Inverted Precision***: $\text{ɥ}_{prec}(p,n) = \frac{N-n}{(P+N)-(p+n)}$
- ***Inverted Laplace***: $\text{ɥ}_{lap}(p,n) = \frac{N-n+1}{(P+N)-(p+n-2)}$
- ***Inverted m-Estimate***: $\text{ɥ}_{mest}(p,n) = \frac{N-n+m\cdot\frac{P}{P+N}}{(P+N)-(p+n-m)}$

Note, however, that some heuristics are insensitive to the difference between top-down refinement and bottom-up selection. For example, *weighted relative accuracy* (WRA) is a heuristic that is frequently used in subgroup discovery [11,13] and has isometrics that are parallel to the diagonal of the coverage space. It is thus equivalent to the simple difference of true positive rate and false negative rate [7]

$$\mathrm{h}_{rdiff}(p,n) = \frac{p}{P} - \frac{n}{N}.$$

The corresponding top-down version would be

$$\text{ɥ}_{rdiff}(p,n) = \frac{N-n}{N} - \frac{P-p}{P}.$$

Obviously ɥ_{rdiff} is equivalent to h_{rdiff} because of

$$\text{ɥ}_{rdiff}(p,n) = \frac{N-n}{N} - \frac{P-p}{P} = \left(1-\frac{n}{N}\right) - \left(1-\frac{p}{P}\right) = \frac{p}{P} - \frac{n}{N} = \mathrm{h}_{rdiff}(p,n).$$

This is also apparent from the isometric structure, which does not change if one switches from a bottom-up version with base $(0,0)$ to a top-down version with base (N,P). However, while frequently used in subgroup discovery, WRA has been shown to over-generalize in a predictive setting [16,9]. We will thus not consider it further in this paper.

3.3 Integration into the Learning Algorithm

One could now think that the new heuristics could be directly plugged into the top-down refinement algorithm of Algorithm 11. However, it is easy to see that this would not yield the desired results. For example, continuing the example of Figure 4, the algorithm would select $c = 1$ as the final refinement for ɥ_{prec}, since all rules covering all positive examples share the same (maximal) heuristic value of 1.0, irrespective of the amount of negative examples they cover. The rule $c = 1 \wedge d = 1$, which covers almost all positive examples but not negative examples, would receive a worse evaluation than its predecessor. Thus, ɥ_{prec} and to a lesser extent ɥ_{lap}, are not well-suited for *rule selection* because rules with high coverage are still preferred by these heuristics, whereas the rule learning process is not steered towards learning a consistent theory. In fact, in preliminary experiments which just replaced the heuristics h_x with their counter-parts ɥ_x so that the latter was used for both rule selection *and* rule refinement, the resulting classifiers were sometimes unable to label *any* new testing example correctly.

Thus, we would like to maintain the conventional heuristics for rule selection, and need to adapt the learning algorithm so that it can use separate heuristics for rule selection and for rule refinement. To realize this, we need to adapt top-down hill-climbing so that different heuristics can be used for rule refinement and rule selection, as marked in the comments of the pseudo-code of Algorithm 11. Algorithm 11 shows the resulting algorithm; lines 2, 6 and 8 have changed.

Algorithm 3. Procedure findBestRule

```
Data: TrainingData
Result: best rule $\mathbf{r}_{best}$
1  $\mathbf{r}_{best} = \emptyset$
2  bestValue = selection_heuristic($\mathbf{r}_{best}$)
3  repeat
4      get possible refinements
5      forall the refinements ref do
6          evaluation = refinement_heuristic(ref)
7      $\mathbf{r}_{ref}$ = best refinement
8      if selection_heuristic($\mathbf{r}_{ref}$) $\geq$ bestValue then
9          $\mathbf{r}_{best} = \mathbf{r}_{ref}$
10 until no refinements left;
11 return $\mathbf{r}_{best}$
```

4 Experiments

In our experimental evaluation, we intend to answer the question whether the proposed separation of rule selection and rule refinement heuristics does indeed yield an improved performance over the standard technique that uses the same heuristic for both tasks.

4.1 Experimental Setup

For our experiments, we use the top-down hill-climbing algorithm implemented in the SECO-library [10] that has also been used in [9]. This is a simple and straight-forward rule learner that solely relies on heuristic rule evaluation to learn a classifier that generalizes well to new data. In particular, no additional procedures, such as pruning or rule optimization, are used to avoid overfitting on the training data. Multiple classes are handled using an ordered one-against-all strategy, as originally proposed for the RIPPER rule learning algorithm [3].

We modified the algorithm as described in Section 3.3, so that it allows for separate criteria for rule selection and rule refinement. We can thus denote an algorithm by a pair $(\mathrm{h}_{\text{selection}}, \mathrm{h}_{\text{refinement}})$. For the experiments we use each of the three standard heuristics for rule selection, and evaluate it with four different heuristics for rule refinement, yielding a total of $3 + 3 \times 3 = 12$ different algorithms. The four heuristics for rule refinement are to use the same heuristic as for rule selection (yielding a standard rule learning algorithm), and to use each of the three inverted heuristics.

For all experiments with h_{mest} and $\mathrm{ɥ}_{mest}$, we used a value of $m = 22.446$ which has been experimentally determined in [9] for the same learning algorithm that forms the basis of our experiments. Thus, this setting is optimized for the use of the m-estimate for guiding both rule selection and refinement. It is most likely a suboptimal value for $\mathrm{ɥ}_{mest}$. However, our main purpose in this paper was not to achieve optimal performance, but to investigate the general properties of different top-down rule refinement heuristics. As we have discussed in Section 2.3 the m-estimate provides a trade-off

Table 1. Number of classes (C), examples (E), and attributes(A) of the 20 datasets used in the experiments

Dataset	C	E	A	**Dataset**	C	E	A
breast-cancer	2	286	10	car	4	1728	7
futebol	2	14	5	contact-lenses	3	24	5
hepatitis	2	155	20	glass	7	214	10
hypothyroid	2	3163	26	idh	3	29	5
horse-colic	2	368	23	iris	3	150	5
ionosphere	2	351	35	lymphography	4	148	19
labor	2	57	17	primary-tumor	22	339	18
mushroom	2	8124	23	monk3	2	122	7
soybean	19	683	36	tic-tac-toe	2	958	10
vote	2	435	17	zoo	7	101	18

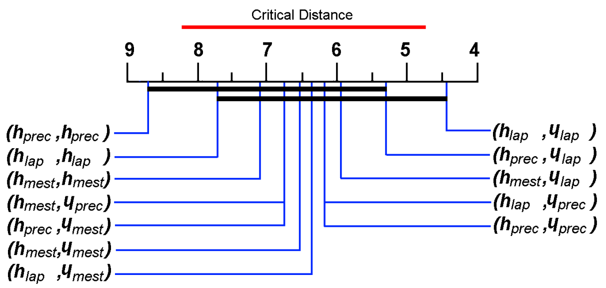


Fig. 6. Nemenyi Test with a significance level of 0.1

between precision and weighted relative accuracy, where larger values of m approach the behavior of WRA, which is insensitive to inversion. In this sense, the m-estimate with $m = 22.446$ nicely complements precision and Laplace in that it is much closer to WRA than the others.

We will evaluate the twelve combinations listed above on 20 datasets by the means of estimated average accuracy. The evaluation method is ten-fold cross-validation to reduce bias and increase the quality of the resulting performance estimate. As can be seen from Table 1, the chosen datasets range from very small datasets (where we feel that good selection heuristics are particularly important) to datasets with several thousand examples. For checking for statistical differences we use Friedman rank tests with a post-hoc Nemenyi test, as recommended by [4].

4.2 Comparison of Average Accuracies

Table 2 shows the detailed results with respect to accuracy. There are three main columns, each corresponding to one of the three rule selection strategies h_{prec}, h_{lap}, and h_{mest}. Each of them has four subcolumns, each corresponding to a rule refinement strategy. The left-most is the standard strategy, and the three others are all three inverted strategies ɥ_{prec}, ɥ_{lap}, and ɥ_{mest}. The best results in each line and each group are underlined.

Table 2. Average accuracies obtained via ten-fold cross-validation on 20 datasets. The best result for each rule selection heuristic is underlined. The bottom line shows the average rank of each rule refinement strategy for each rule selection heuristic.

	$(\mathrm{h}_{prec}, .)$				$(\mathrm{h}_{lap}, .)$				$(\mathrm{h}_{mest}, .)$			
Dataset	h_{prec}	ɥ_{prec}	ɥ_{lap}	ɥ_{mest}	h_{lap}	ɥ_{prec}	ɥ_{lap}	ɥ_{mest}	h_{mest}	ɥ_{prec}	ɥ_{lap}	ɥ_{mest}
breast-cancer	68.53	72.38	72.03	$\underline{73.43}$	69.58	70.63	71.33	$\underline{72.73}$	71.33	72.03	72.38	$\underline{73.78}$
car	90.10	90.34	$\underline{90.51}$	88.66	90.45	91.20	$\underline{91.73}$	91.20	89.64	$\underline{90.45}$	90.28	87.91
contact-lenses	79.17	$\underline{87.50}$	$\underline{87.50}$	83.33	79.17	$\underline{87.50}$	$\underline{87.50}$	83.33	$\underline{87.50}$	$\underline{87.50}$	$\underline{87.50}$	83.33
futebol	28.57	$\underline{64.29}$	57.14	42.88	28.57	$\underline{64.29}$	57.14	42.88	50.00	$\underline{64.29}$	57.14	42.86
glass	56.54	65.89	$\underline{68.69}$	62.15	61.22	65.89	$\underline{68.69}$	62.15	69.16	67.29	$\underline{71.50}$	63.55
hepatitis	78.07	79.36	$\underline{80.00}$	76.77	78.71	79.36	$\underline{80.00}$	76.74	78.07	79.36	$\underline{80.00}$	76.77
hypothyroid	98.23	98.61	98.74	$\underline{98.83}$	98.39	98.61	98.74	$\underline{98.83}$	98.80	98.61	98.74	$\underline{98.83}$
horse-colic	72.01	$\underline{79.35}$	$\underline{79.35}$	77.99	70.65	79.35	$\underline{80.16}$	77.99	77.45	$\underline{79.35}$	78.80	77.99
idh	62.07	$\underline{82.76}$	75.86	75.86	62.07	$\underline{82.76}$	75.86	75.86	68.97	$\underline{82.76}$	75.86	75.86
iris	92.67	93.33	$\underline{95.33}$	94.67	94.00	93.33	$\underline{95.33}$	94.67	94.00	93.33	$\underline{95.33}$	94.67
ionosphere	$\underline{95.16}$	82.62	83.19	89.46	$\underline{94.87}$	82.62	93.19	89.46	$\underline{91.74}$	82.91	83.19	91.17
labor	$\underline{91.23}$	80.70	82.46	89.47	$\underline{91.23}$	80.70	82.46	89.47	85.97	80.70	82.46	$\underline{89.47}$
lymphography	83.78	77.70	$\underline{84.46}$	83.11	$\underline{85.14}$	77.70	84.46	83.11	75.00	76.35	81.08	$\underline{83.78}$
mushroom	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
monk3	$\underline{87.71}$	82.79	82.79	84.43	$\underline{88.53}$	85.25	84.43	86.89	81.15	79.51	81.15	$\underline{82.79}$
primary-tumor	33.63	$\underline{39.23}$	35.10	30.97	32.45	$\underline{39.23}$	35.99	30.38	33.92	$\underline{37.76}$	34.51	30.68
soybean	90.04	91.51	$\underline{92.24}$	91.36	90.34	91.80	$\underline{92.39}$	90.63	$\underline{91.51}$	90.92	90.48	91.36
tic-tac-toe	97.39	$\underline{98.02}$	97.60	97.81	97.60	$\underline{98.02}$	97.60	97.91	$\underline{98.12}$	98.02	97.60	97.81
vote	$\underline{94.94}$	93.56	94.25	94.48	$\underline{95.40}$	94.25	94.25	94.94	93.33	93.56	94.71	$\underline{96.09}$
zoo	84.16	88.12	$\underline{92.08}$	90.01	86.14	88.12	$\underline{92.08}$	90.10	89.11	88.12	$\underline{92.08}$	90.10
average rank	3.075	2.400	$\underline{1.975}$	2.550	3.000	2.500	$\underline{1.975}$	2.525	2.700	2.625	$\underline{2.225}$	2.450

Not surprisingly, we can see that each of the combinations works best in some cases, and that the differences can be quite large in some cases (mostly for rather small datasets). In order to get a better overall impression, we show the average ranks within each group in the last line of the table. This gives a fairly consistent picture in that the standard strategy always performs worst, i.e., on average all three inverted rule refinement heuristics perform better than the case where rule refinements are evaluated with the same heuristic as rule refinements. Thus, our expectation that top-down refinement heuristics work better than conventional heuristics has been confirmed.

The results are also consistent in that the inverted Laplace-heuristic always performs best for all rule selection strategies, whereas the m-estimate and precision are about equal on ranks 2 and 3. We can also see that the differences between the methods are much smaller for the m-estimate than for the others. In fact, a Friedman test reveals that the results within the h_{prec} rule selection group are statistically significant at a 5% level, the results within the h_{lap} group at the 10% level, whereas the results in the m-estimate are only weakly different. However, this is not surprising, because as we noted above, for larger values of m, the behavior of the m-estimate approaches the behavior of WRA, which is insensitive to inversion. Thus, with increasing values of m, the results must become more and more similar to each other.

Table 3. Comparison of the number of rules (R) and conditions (L) for regular and inverted Laplace heuristics

	$(\mathrm{h}_{lap}, \mathrm{h}_{lap})$		$(\mathrm{h}_{lap}, \mathrm{h}'_{lap})$			$(\mathrm{h}_{lap}, \mathrm{h}_{lap})$		$(\mathrm{h}_{lap}, \mathrm{h}'_{lap})$	
Dataset	R	L	R	L	**Dataset**	R	L	R	L
breast-cancer	25	67	38	173	ionosphere	17	25	8	42
car	107	495	107	506	labor	5	7	3	12
contact-lenses	5	14	5	15	lymphography	18	42	11	47
futebol	4	7	2	5	monk3	13	38	11	32
glass	50	103	14	83	mushroom	11	13	7	35
hepatitis	13	26	7	46	primary-tumor	80	319	72	518
horse-colic	44	114	19	111	soybean	62	134	45	195
hypothyroid	27	65	9	69	tic-tac-toe	22	84	16	69
iris	7	15	5	17	vote	13	48	12	58
idh	4	5	2	5	zoo	19	19	6	14
averages						27.3	82.0	20.0	102.6

4.3 Validation and Algorithm Comparison

Overall, the combination $(\mathrm{h}_{lap}, \text{ɥ}_{lap})$ outperforms other combinations on seven datasets (namely *car, contact-lenses, hepatitis, horse-colic, iris, soybean* and *zoo*). As such, this combination in particular becomes interesting for further validation. We will now conduct statistical tests to try and prove the assumption that the combination $(\mathrm{h}_{lap}, \text{ɥ}_{lap})$ is superior w.r.t. accuracy.

Using $N = 20$ datasets with $k = 12$ algorithms, we obtain a chi-square value of 19.792 and a corresponding F_F statistic of 1.878. The corresponding critical value based on a significance level of 0.05 with 11 and 209 degrees of freedom is 1.834, resulting in a passed Friedman test (failure at level 0.01). The ranks of the algorithms as well as the critical distance for the post-hoc Nemenyi test are shown in Figure 6. Although the results only show that the combination $(\mathrm{h}_{lap}, \text{ɥ}_{lap})$ is significantly better than the algorithm $(\mathrm{h}_{prec}, \mathrm{h}_{prec})$, which is known to overfit the data, it is still remarkable that all combinations that involve an inverted heuristic are higher-ranked than all three original heuristics, including $(\mathrm{h}_{mest}, \mathrm{h}_{mest})$, which was one of the best-performing algorithms in a previous study [9].

4.4 Number of Rules and Conditions

Using inverted heuristics also has an effect on the nature of conditions that are selected. In short, whereas regular heuristics focus mostly on consistency, inverted heuristics tend to add conditions that maintain completeness. For example, if at any point, both heuristics are faced with the choice of adding an incomplete but consistent rule $\mathbf{r}_1$ (a point on the P-axis) and a complete but inconsistent rule $\mathbf{r}_2$ (a point (P, n) for some value $0 < n < N$), regular precision would give a maximum evaluation of $\mathrm{h}_{prec}(\mathbf{r}_1) = 1.0$ to $\mathbf{r}_1$, whereas inverted precision gives a maximum score $\text{ɥ}_{prec}(\mathbf{r}_2) = 1.0$ to $\mathbf{r}_2$. This has the effect that inverted heuristics bias the learner towards conditions that do not add additional discriminative power (but are nevertheless informative).

```
2160  p :- odor = f.
1152  p :- gill-color = b.
 256  p :- odor = p.
 192  p :- odor = c.
  72  p :- spore-print-color = r.
  36  p :- stalk-color-below-ring = c.
  24  p :- stalk-color-below-ring = y.
   4  p :- cap-surface = g.
   1  p :- cap-shape = c.
  16  p :- stalk-color-below-ring = n, stalk-surface-above-ring = k.
   3  p :- habitat = l, stalk-color-below-ring = w.
```

(a) using h_{lap} for refinement

```
2192  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.
```

(b) using ꟼ_{lap} for refinement

Fig. 7. Decision lists learned for the class `poisonous` in the *mushroom* dataset, along with the number of positive examples covered by each rule (no rule covers any negative examples)

In practical terms, inverted heuristics tend to learn longer rules, which will nevertheless, somewhat counter-intuitively, have a higher coverage than those learned with regular heuristics. As an illustration, Table 3 compares the number of rules and conditions induced with h_{lap} to those induced with ꟼlap, the latter being the the configuration that achieved the best results in our experiments. On 17 out of 20 datasets the inverted version learns a lower number of rules, on two an equal number of rules, and only on one dataset a higher number of rules, which clearly confirms that the learned rules on average tend to have a higher coverage. Moreover, on 13 datasets ꟼ_{lap} has a higher number of conditions, on one dataset it is equal and on 6 the number is smaller, which confirms that the rules learned by inverted heuristics tend to be longer. Both findings are also confirmed by the averages shown in the last line of Table 3.

Note, however, that this does not necessarily reduces the comprehensibility of the learned rules. In a way, in the terminology of Michalski [14], inverted heuristics tend to find *characteristic* descriptions, whereas standard heuristics tend to find *discriminative* descriptions. As an illustration, Figure 7 shows the rule sets learned for the *mushroom* dataset. Both rule sets cover all 3196 examples of poisonous mushrooms. However, while the first rule set, learned with a traditional heuristic, focuses on single characteristics such as the odor of the mushroom, the second rule set contains much more descriptive rules. Interestingly, the used attributes are quite different (e.g., `odor` does not appear at all in the latter rule set). Another interesting observation is that the former rule set contains some rules with only very low coverage: the last six rules all cover fewer examples than the last rule of the rule set learned with the inverted heuristic. One reason for this is that because the previous rules are somewhat less general, they also leave more examples to be classified for subsequent rules. For example, the last rule of Figure 7 (b) still classifies 44 examples, whereas a similar rule that consisting only of the first condition has only 24 examples left to classifier in Figure 7 (a).

We are not expert enough to judge the plausibility of the rules of Figure 7, but in general, we think that more detailed rules can be more convincing and are certainly no less comprehensible than the general discriminative rules. In fact, we think that this property of inverted heuristics is also of particular interest to subgroup discovery [12], although we leave this as subject for future work.

5 Conclusions and Open Questions

In this paper, we made two contributions to heuristic inductive rule learning. First, we argued that it may be beneficial to separate the evaluation of candidates for rule refinement and the selection of rules for the final theory. Accordingly, we suggest to use different criteria for both. Second, we showed that conventional precision-based heuristics can be inverted in the sense that they do not evaluate candidate refinements from the point of view of the origin of the coverage space, but from the point of view of their predecessor rule in a top-down search. Our experiments showed that the use of such inverted heuristics for evaluating rule refinements leads to better results than the use of the original versions. Interestingly, inverted heuristics also have the tendency to learn rule sets with longer but fewer rules.

Our results are so far confined to top-down covering rule learning algorithms. While we do not expect that bottom-up algorithms would profit from inverted heuristics, which reflect a top-down search strategy, it remains an open question whether other top-down algorithms may benefit from their use. In particular, the fact that inverted heuristics tend to learn longer, characteristic rules may be of interest for subgroup discovery.

We have also only considered precision-like heuristics in this work, mainly because the m-estimate has delivered a state-of-the-art performance in a large comparative study [9], so that it seemed a natural point of departure for our experiments. While we have shown that other linear heuristics such as WRA, which is popular in subgroup discovery, cannot be inverted, we still need to look at heuristics with non-linear isometrics. In particular, the proposed separation of rule refinement and rule selection criteria is also closely related to the use of pruning criteria, which filter out unpromising rules. It remains to be seen whether conventional rule pruning criteria, such as the significance test of CN2 [2,1] may also be used favorably as rule selection criteria. Furthermore, we also deliberately refrained from optimizing the m-parameter of the inverted heuristics in any way because we wanted to avoid to obtain good results for the inverted heuristics that are only due to an extensive search for optimal parameter values. However, such an evaluation is planned as the next step in our work.

Finally, we note that the use of precision for rule selection may be viewed as a simple, greedy maximization of the area under the ROC curve (AUC) [7]. The inverted precision heuristic introduced in this paper may be viewed as a counter-part that maximizes the AUC for individual rules. Interestingly, in preliminary experiments we could not demonstrate that improving the AUC maximization for individual rules also leads to a better AUC for the entire theory. However, this needs a deeper investigation and needs to be put into perspective with alternative approaches to maximize the AUC in inductive rule learning [15,5].

Acknowledgements. The authors would like to thank the anonymous reviewers for their comments, which helped to improve this paper.

References

1. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In: Proceedings of the 5th European Working Session on Learning (EWSL 1991), Porto, Portugal, pp. 151–163. Springer, Heidelberg (1991)
2. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning 3(4), 261–283 (1989)
3. Cohen, W.W.: Fast effective rule induction. In: Prieditis, A., Russell, S. (eds.) Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, July 9-12, vol. 123, pp. 115–123. Morgan Kaufmann (1995)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
5. Fawcett, T.E.: PRIE: A system for generating rulelists to maximize ROC performance. Data Mining and Knowledge Discovery 17(2), 207–224 (2008)
6. Fürnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review 13(1), 3–54 (1999)
7. Fürnkranz, J., Flach, P.A.: ROC 'n' rule learning – Towards a better understanding of covering algorithms. Machine Learning 58(1), 39–77 (2005)
8. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer, Heidelberg (2012)
9. Janssen, F., Fürnkranz, J.: On the quest for optimal rule learning heuristics. Machine Learning 78(3), 343–379 (2010)
10. Janssen, F., Zopf, M.: The SeCo-framework for rule learning. In: Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA (2012)
11. Klösgen, W.: Explora: A multipattern and multistrategy discovery assistant. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 249–271. AAAI Press (1996)
12. Kralj Novak, P., Lavrač, N., Webb, G.I.: Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. Journal of Machine Learning Research 10, 377–403 (2009)
13. Lavrač, N., Kavšek, B., Flach, P., Todorovski, L.: Subgroup discovery with CN2-SD. Journal of Machine Learning Research 5, 153–188 (2004)
14. Michalski, R.S.: A theory and methodology of inductive learning. Artificial Intelligence 20(2), 111–162 (1983)
15. Prati, R.C., Flach, P.A.: Roccer: An algorithm for rule learning based on ROC analysis. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, Scotland, pp. 823–828. Professional Book Center (2005)
16. Todorovski, L., Flach, P.A., Lavrač, N.: Predictive performance of weighted relative accuracy. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 255–264. Springer, Heidelberg (2000)