Mach Learn (2010) 78: 343–379
DOI 10.1007/s10994-009-5162-2

On the quest for optimal rule learning heuristics

Frederik Janssen · Johannes Fürnkranz

Received: 14 March 2008 / Revised: 2 November 2009 / Accepted: 4 November 2009 /
Published online: 9 December 2009

Abstract The primary goal of the research reported in this paper is to identify what criteria are responsible for the good performance of a heuristic rule evaluation function in a greedy top-down covering algorithm. We first argue that search heuristics for inductive rule learning algorithms typically trade off consistency and coverage, and we investigate this trade-off by determining optimal parameter settings for five different parametrized heuristics. In order to avoid biasing our study by known functional families, we also investigate the potential of using metalearning for obtaining alternative rule learning heuristics. The key results of this experimental study are not only practical default values for commonly used heuristics and a broad comparative evaluation of known and novel rule learning heuristics, but we also gain theoretical insights into factors that are responsible for a good performance. For example, we observe that consistency should be weighted more heavily than coverage, presumably because a lack of coverage can later be corrected by learning additional rules.

Keywords Inductive rule learning · Heuristics · Metalearning

1 Introduction

The long-term goal of our research is to understand the properties of rule learning heuristics, that will allow them to perform well in a wide variety of datasets. Although different classification rule learning algorithms use different heuristics, there has not been much work on trying to characterize their behavior. Notable exceptions include Lavrač et al. (1999), which proposed weighted relative accuracy as a novel heuristic, and Fürnkranz and Flach (2005), in which a wide variety of rule evaluation metrics were analyzed and compared by

Editor: Hendrik Blockeel.

F. Janssen · J. Fürnkranz (✉)
Technische Universität Darmstadt, Darmstadt, Germany
e-mail: juffi@ke.informatik.tu-darmstadt.de

F. Janssen
e-mail: janssen@ke.informatik.tu-darmstadt.de

visualizing their behavior in ROC space. There are also some works on comparing properties of association rule evaluation measures (e.g., Tan et al. 2002) but these have different requirements than classification rules (e.g., completeness is not an issue there).

In this paper, we will try to approach this problem empirically. We will first empirically compare and analyze a number of known rule learning heuristics. Rule learning heuristics, in one way or another, trade off consistency and coverage. On the one hand, rules should be as consistent as possible by only covering a small percentage of negative examples. On the other hand, rules with a high coverage tend to be more reliable, even though they might be less precise on the training examples than alternative rules with lower coverage. An increase in coverage of a rule typically goes hand-in-hand with a decrease in consistency, and vice versa. In fact, the conventional top-down hill-climbing search for single rules follows exactly this principle: starting with the empty rule, conditions are greedily added, thereby decreasing coverage but increasing consistency.

In this work, we will show that five well-known rule evaluation metrics (a cost trade-off, a relative cost trade-off, the m-estimate, the F-measure, and the Klösgen measures) provide parameters that allow to control this trade-off. In an extensive experimental study—to our knowledge the largest empirical comparison of rule learning heuristics to date—we aimed at determining optimal values for each of their respective parameters. We will compare these settings to standard heuristics and show that the new settings outperform the fixed consistency/coverage trade-offs that are commonly used as rule learning heuristics. By testing the performance of the optimized heuristics on an additional selection of datasets not used for optimization, we will ensure that this performance gain is not due to overfitting the training datasets.

However, optimizing parameters constrains the candidate heuristics to known functional shapes. Consequently, we will then try to leave these constraints behind and try to discover entirely new heuristics. The key idea is to meta-learn such a heuristic from experience, without a bias towards existing measures. Consequently, we created a large meta dataset (containing information from which we assume that the "true" performance of a rule can be learned) and use various regression methods to learn to predict this performance. On this dataset, we learn an evaluation function and use it as a search heuristic inside our implementation of a simple rule learner. We report on the results of our experiments with various options for generating the meta datasets, with different feature sets and different metalearning algorithms. In particular, we try to evaluate the importance of rule length as an additional feature and consider a delayed-reward scenario where the learner tries to predict the performance of the *completed* rule from its incomplete current state in the search space.

The paper is organized as follows: we start with a brief recapitulation of separate-and-conquer learning and describe our simple ruler learner, which is used for generating the meta data and for evaluating the learned heuristics (Sect. 2). Section 3 then provides a survey of the heuristics that are experimentally compared in this paper. In this section, we also briefly recapitulate the use of coverage space isometrics for visualizing the preference structure of rule learning heuristics. After a brief description of the experimental setup that will be used throughout the paper (Sect. 4), the main part of the paper describes our experimental work in optimizing known heuristics (Sect. 5) and metalearning new heuristics (Sect. 6). We put the results in perspective by discussing which problems need to be addressed via rule learning heuristics, and which of them are addressed in this work (Sect. 7). The paper is wrapped up with a brief discussion of related work (Sect. 8) and a summary of the most important conclusions drawn from this study (Sect. 9).

Parts of this paper have previously appeared in Janssen and Fürnkranz (2008) and Janssen and Fürnkranz (2007).

2 Separate-and-conquer rule learning

The goal of an inductive rule learning algorithm is to automatically learn a ruleset from a given dataset that allows to map unseen examples on their correct classes. Algorithms differ in the way they learn individual rules, but most of them employ a *separate-and-conquer* or *covering* strategy for combining rules into a rule set (Fürnkranz 1999). The origin of this strategy is the AQ-Algorithm (Michalski 1969) but it is still used in many algorithms, most notably in Ripper (Cohen 1995), arguably one of the most accurate rule learning algorithms today.

2.1 The basic algorithm

Separate-and-conquer rule learning can be divided into two main steps: First, a single rule is learned from the data (the *conquer* step). Then all examples which are covered by the learned rule are removed from the training set (the *separate* step), and the remaining examples are "conquered". The two steps are iterated until no more positive examples are left. In the simplest version, this ensures that every positive example is covered at least by one rule (*completeness*) and no negative example is included (*consistency*). More complex versions of the algorithm will allow certain degrees of incompleteness (leaving some examples uncovered) and inconsistencies (covering some negative examples). In the remainder of the paper, we will use the terms *completeness* and *consistency* to denote these gradual concepts.

For the purpose of this empirical study, we implemented a simple separate-and-conquer or covering rule learning algorithm within the SeCo-Framework, a modular architecture for rule learning, which is currently under development at our group (Fürnkranz 1999; Thiel 2005). Both the covering algorithm and the top-down refinement inside the covering loop are fairly standard. However, covering algorithms often differ in details, so we believe it is worth-while to specify exactly how we proceeded.

Algorithm 1 shows the basic covering loop. It repeatedly learns one rule by calling GREEDYTOPDOWN, removes all examples covered by this rule from the training set, and adds the rule to the final theory. This is repeated until no more positive examples are left or until adding the best learned rule would not increase the accuracy of the rule set on the training set (which is the case when the rule covers more negative than positive examples).

Algorithm 1 SEPARATEANDCONQUER*(Examples)*

```
# loop until all positive examples are covered
Theory ← ∅
while POSITIVE(Examples) ≠ ∅
    # find the best rule
    Rule ← GREEDYTOPDOWN(Examples)
    # stop if it doesn't cover more positives than negatives
    if |COVERED(Rule, POSITIVE(Examples))|
        ≤ |COVERED(Rule, NEGATIVE(Examples))|
        break
    # remember rule and remove covered examples
    Theory ← Theory ∪ Rule
    Examples ← Examples \ COVERED(Rule,Examples)
return Theory
```

Algorithm 2 GREEDYTOPDOWN*(Examples)*

```
# remember the rule with the best evaluation
BestRule ← MaxRule ← null
BestEval ← EVALUATERULE(BestRule,Examples)

do
    # compute refinements of the best previous rule
    Refinements ← REFINEMENTS(MaxRule)

    # find the best refinement
    MaxEval ← −∞
    for Rule ∈ Refinements
        Eval ← EVALUATERULE(Rule,Examples)
        if Eval > MaxEval
            MaxRule ← Rule
            MaxEval ← Eval

    # store the rule if we have a new best
    if MaxEval ≥ BestEval
        BestRule ← MaxRule
        BestEval ← MaxEval

# break loop when no more refinements
until Refinements = ∅

return BestRule
```

Algorithm 2 shows the basic algorithm for learning a single rule with greedy top-down search. The algorithm starts with an initially empty rule (a rule that covers all examples). The rule is successively refined by adding conditions to its body. Conditions are either tests for equality with a specific value of a discrete attribute, or, in the case of a continuous attribute, a comparison ($<$ or $\geq$) with a threshold value (half-way between two adjacent values of the training data). All candidate refinements are evaluated with a heuristic EVALUATERULE, and the best refinement is stored in *MaxRule*. It is then checked whether *MaxRule* is better than the current best rule, and the procedure recursively continues with the refinements of *MaxRule*. If no further refinements are possible, the search stops and the best rule encountered during the search is returned.

Our implementation of the algorithm made use of a few optimizations that are not shown in Algorithm 2. Among them are stopping the refinement process when no more negative examples are covered, random tie breaking for rules with equal heuristic evaluations, and filtering out candidate rules that do not cover any positive examples (this may make a huge difference in the number of rules generated for the accuracy heuristic). To speed up the implementation, we also stop searching the refinements of a rule if its best possible refinement—the virtual rule that covers all remaining positive examples and none of the remaining negative examples—has a lower evaluation than the current best rule.

The algorithm shown here only works for concept learning problems with positive and negative training examples. Multi-class problems are tackled using the *ordered class binarization* that has been suggested for the Ripper rule learner (Cohen 1995): First, the classes are sorted according to ascending frequency of occurrence in the training data. Then the algorithm successively learns rules for the i-th class, using the examples of this class as the

positive examples and the examples of all classes $j > i$ as the negative examples. Covered examples are removed from the training set. No rules are learned for the final, the largest class, but instead a default rule is added that always predicts this class when no other rule fires. At classification time, the learned rules are interpreted as a decision list, i.e., the class of the first rule that fires is predicted.

2.2 Discussion of the algorithm

We want to stress that our algorithm is quite typical for commonly used covering algorithms. In particular, it is more or less identical to the second version of the popular CN2 (Clark and Boswell 1991) algorithm. The main difference lies in the class binarization. CN2 can be used in two different modes: an *unordered* mode, which learns rules for each class, always using all other classes as the negative examples, and a *decision-list* mode, which is able to learn rule lists with arbitrary class assignments. For example the first rule may predict the first class, the second one the second class and the third one again the first class. While the ordered class binarization used in our implementation also learns decision lists, they are less flexible in that the order of the classes in the list is fixed and rules of different classes may not alternate. Other differences include that CN2 is not able to handle missing class values, and that it uses a beam search with beam width 5 by default (our implementation uses Hill-Climbing search, i.e., the beam is set to 1). CN2 also employs rule filtering, which removes redundant rules and is not used in our algorithm.

Because we wanted to gain a principled understanding of what constitutes a good evaluation metric for inductive rule learning, we did not employ explicit stopping criteria or pruning techniques for overfitting avoidance, but solely relied on the evaluation of the rules by the used rule learning heuristic. Note, however, that this does not necessarily mean that we learn an overfitting theory that is complete and consistent on the training data (i.e., a theory that covers all positive and no negative examples), because many heuristics will prefer impure rules with a high coverage over pure rules with a lower coverage. This was already noted by the authors of CN2, who observed that the importance of its rule significance test greatly diminished when Laplace is used as a search heuristic because, compared to entropy, it tends to favor general rules anyways (Clark and Boswell 1991). In fact, in the README file to the implementation of the algorithm, one can read that the Laplace heuristic of the second version directly tries to estimate what the combination of entropy and significance test indirectly estimated, namely the expected performance of a rule on new test data, and that thus the Laplace heuristic is intended to replace both, the original entropy heuristic and the significance test. In our version of the algorithm, the choice of the learning heuristic is an additional parameter, and in the following, we will try to understand what would constitute a good choice for it.

Our algorithm is also quite similar to the Foil (Quinlan 1990) algorithm, which forms the basis of many rule learning algorithms, most notably Ripper (Cohen 1995). The key difference here is that Foil-based algorithms do not evaluate refinements on an absolute scale, but relative to their respective predecessors, i.e., they focus on the *gain* that a rule obtains in comparison to its predecessor. While this is a reasonable approach, gain-based algorithms can not directly compare the evaluation of two rules with different predecessors, and are therefore not able to identify the best rule encountered during the search. Instead, they always return the last rule searched. Thus, their performance crucially depends on the availability of a pruning heuristic or a stopping criterion, which determines when the refinement process should stop. Foil uses a heuristic based on minimal description length for this purpose (Quinlan 1990; Fürnkranz and Flach 2004), whereas Ripper employs the incremental reduced error pruning technique, which prunes each rule after it has been learned

(Fürnkranz and Widmer 1994; Fürnkranz 1997). On the other hand, algorithms of the type shown in Algorithm 2 do not necessarily return the last rule searched, but the rule with the highest evaluation encountered during the search. In this case, a stopping heuristic assumes the role of a filtering criterion, which filters out unpromising candidates, but does not directly influence the choice of the best rule (Clark and Boswell 1991). Because of this dependency on stopping criteria, we do not further consider gain-based heuristics in this paper. However, we note that an empirical study comparing gain-based to absolute heuristics is an open research question.

3 Rule learning heuristics

The goal of a rule learning algorithm is to find a simple set of rules that explains the training data and generalizes well to unseen data. This means that individual rules have to optimize two criteria simultaneously:

Coverage: the number of positive examples that are covered by the rule should be maximized and

Consistency: the number of negative examples that are covered by the rule should be minimized.

Thus, each rule can be characterized by

- p and $n \equiv$ the positive/negative examples covered by the rule
- P and $N \equiv$ the total number of positive/negative examples in the training set

Consequently, most rule learning heuristics depend on p, n, P, and N, but combine these values in different ways.

A few heuristics also include other parameters, such as

- $l \equiv$ the length of the rule and
- p' and $n' \equiv$ the number of positive and negative examples that are covered by the rule's predecessor

Later on in this paper, we will evaluate the utility of taking the rule's length into account (cf. Sect. 6.2.2). However, as our goal is to evaluate a rule irrespective of how it has been learned, we will not consider the parameters p' and n'. Heuristics like Foil's information gain (Quinlan 1996), which include p' and n', may yield different evaluations for the same rule, depending on the order in which its conditions have been added to the rule body. Moreover, as discussed above (Sect. 2.2), rules with different predecessors are not comparable, and thus it is not possible to return the best rule encountered in a search. We will not further consider heuristics of this type in this paper.

As P and N are constant for a given dataset, heuristics differ effectively only in the way they trade off completeness (maximizing p) and consistency (minimizing n). Thus they may be viewed as functions $h(p, n)$. We will denote rule evaluation heuristics by the letter h with a subscript to differentiate between them. As all heuristics depend only on the number of covered positive and negative examples, they are unable to discriminate between rules that cover the same number of positive and negative examples. So it follows from the first observation that $h(R_i) \equiv h(n_i, p_i)$ holds for all rules R_i. Resulting from the second observation it is obvious that $R_1 \neq R_2 \nrightarrow h(R_1) \neq h(R_2)$.

In the following, we will survey the heuristics that will be investigated in this paper. Most (but not all) of these heuristics have already been discussed by Fürnkranz and Flach (2005), so we will keep the discussion short. We discriminate between *basic heuristics* (Sect. 3.2), which primarily focus on one aspect, *composite heuristics* (Sect. 3.3), which provide a fixed trade-off between consistency and coverage, and *parametrized heuristics* (Sect. 3.4), which provide a parameter that allows to tune this trade-off. However, first we will briefly recapitulate coverage spaces, which will be our primary means of visualizing the behavior of the investigated heuristics.

3.1 Visualization with coverage space isometrics

Fürnkranz and Flach (2005) suggested to visualize the behavior of rule learning heuristics by plotting their isometrics in *coverage space*, an un-normalized version of ROC-space. Unlike ROC-spaces, the coverage space plots p (the absolute number of covered positive examples) on the y-axis and n (the absolute number of covered negatives) on the x-axis. For example the point $(0, 0)$ represents the empty theory where no example is covered at all. A good algorithm should navigate the learning process in the direction of the point $(0, P)$, which represents the optimal theory that covers all positive examples and no negatives. The point $(N, 0)$ represents the opposite theory, and the universal theory, covering all P positive and N negative examples, is located at (N, P).

We can also represent individual rules R_i by a point (n_i, p_i) where $n_i \in N$ are the covered negative examples and $p_i \in P$ are the covered positives. *Isometrics* connect rules $R_1, \ldots, R_m$ which have an identical heuristic value but cover different numbers of examples. The preference bias of different heuristics may then be visualized by plotting the respective heuristic values of the rules on top of their locations in coverage space, resulting in a 3-dimensional (3-d) plot $(p, n, h(p, n))$ (right picture of Fig. 1). A good way to view this graph in two dimensions is to plot the *isometrics* of the learning heuristics, i.e., to show contour lines that connect rules with identical heuristic evaluation values. Figure 1 shows examples of a 2-d and 3-d coverage space that both contain isometrics of accuracy $(p - n)$. The left one shows the respective values assigned by the heuristic as numbers attached to the contour lines whereas the right one shows them as a 3-d surface. The rules R1 (covering 15 negatives and 25 positives) and R2 ($n = 25,\ p = 35$) both have an accuracy of 10 and therefore lie on the same isometric. For visualization, one is primarily interested in the shape of the isometrics. Thus, we will typically omit the evaluation value from the graph and prefer the 2-d plots.

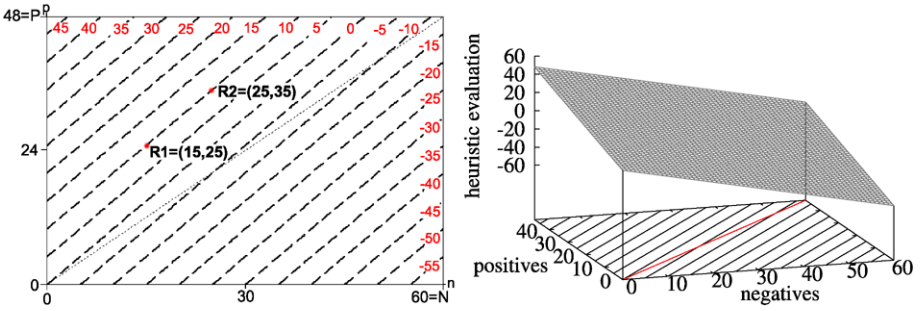


Fig. 1 Isometrics in 2-d and 3-d coverage space

3.2 Basic heuristics

These heuristics are rather simple and do either optimize consistency or coverage on its own.

- *true positive rate (recall)* $h_{\text{tpr}} = h_{\text{rec}} = \frac{p}{P}$

computes the coverage on the positive examples only. It is—on its own—equivalent to simply using p (because P is constant). Due to its independence of covered negative examples, its isometrics are parallel horizontal lines.

- *false positive rate* $h_{\text{fpr}} = \frac{n}{N}$

computes the coverage on the negative examples only. Its isometrics are parallel vertical lines.

- *full coverage* $h_{\text{cov}} = \frac{p+n}{P+N}$

computes the fraction of all covered examples. The maximum heuristic value is reached by the universal theory, which covers all examples (the point (N, P) of the coverage space). The isometrics are parallel lines with a slope of -1 (similar to those of the lower right graph in Fig. 3).

3.3 Composite heuristics

The heuristics shown in the previous section only optimize one of the two criteria, consistency or coverage. In this section, we will discuss a few standard heuristics that provide a fixed trade-off between consistency and coverage.

- *precision* $h_{\text{prec}} = \frac{p}{p+n}$

computes the fraction of correctly classified examples (p) among all covered examples ($p + n$). Its isometrics are rotating around the origin. Precision is known to learn overly complex rules, as will also become obvious from the results shown in Tables 2 and 3. More precisely, for rules with high consistency, coverage becomes less and less important. All rules with maximum consistency ($h_{\text{prec}} = 1.0$) are considered to be equal, irrespective of their coverage. This can be seen nicely from the isometric structure, where the slopes of the isometrics become steeper and steeper when they approach the P-axis, which by itself forms the isometric for the maximum consistency case. The inverse behavior (preferring coverage over consistency for regions with high coverage) can also be observed near the N-axis, but this region is not interesting for practical rule learning systems.

- *Laplace* $h_{\text{Lap}} = \frac{p+1}{p+n+2}$

is an attempt to alleviate the overfitting behavior of h_{prec} by initializing the counts for p and n with 1, thereby effectively moving the rotation point of precision to $(-1, -1)$ in the coverage space. It is used in the CN2-algorithm (Clark and Niblett 1989). However, it is known that the Laplace heuristic will still lead to serious overfitting if used without appropriate pruning heuristics. Thus, it also places too strong emphasis on consistency over coverage.

- *accuracy* $h_{\text{acc}} = p - n$

computes the percentage $(p + (N - n))/(P + N)$ of correctly classified examples among all training examples. As P and N are typically constant for the evaluation of a set of candidate rules, this is equivalent to the simpler $p - n$. Its isometrics in coverage space are parallel lines with a slope of 1 (45 degrees) as depicted in Fig. 1. Accuracy has been used

as a pruning criterion in I-REP (Fürnkranz and Widmer 1994), and (with a penalty on rule length) as a selection criterion in Progol (Muggleton 1995). We will see later in this paper that this measure over-generalizes, i.e., it places too strong emphasis on coverage.

- *weighted relative accuracy (WRA)* $h_{\text{WRA}} = h_{\text{tpr}} - h_{\text{fpr}}$

computes the difference between the true positive rate and the false positive rate. The basic idea of *weighted relative accuracy* (Lavrač et al. 1999) is to compute accuracy on a normalized distribution of positive and negative examples. As a result, the lines of the isometrics are now parallel to the diagonal of the coverage space instead of those of h_{acc} which have a slope of 1 (cf. upper right graph of Fig. 3). The measure has been successfully used in subgroup discovery (Lavrač et al. 2004). However, for inductive rule learning, the experimental evidence of Todorovski et al. (2000), which is consistent with our own experience presented later in this paper, suggests that this measure has a tendency to overgeneralize.

- *correlation* $h_{\text{corr}} = \frac{pN - nP}{\sqrt{P \cdot N \cdot (p+n) \cdot (P-p+N-n)}}$

computes the correlation coefficient between the predicted and the target labels. Like h_{WRA}, its isometrics are symmetrical around the diagonal, but their ends are bended towards the $(0, 0)$ and (N, P) points. The measure has exhibited a very good performance in the inductive rule learning algorithm Fossil (Fürnkranz 2004) (where it was formulated as a Foil-type gain heuristic, i.e., p' and n' were used instead of P and N), and has been frequently used in association rule and subgroup discovery (Brin et al. 1997; Xiong et al. 2004).

3.4 Parametrized heuristics

Although the measures discussed in the previous section aim at trading off consistency and coverage, they implement a fixed trade-off, which, as experience shows, is not optimal, e.g., it often unduly prefersconsistency or coverage. In this section, we will discuss five heuristics that allow to tune this trade-off with a parameter. We will start with two cost measures, which directly trade off absolute or relative positive and negative coverage. Thereafter, we will see three measures that use h_{prec} for optimizing consistency, but use different measures $(h_{\text{rec}}, h_{\text{WRA}}, h_{\text{cov}})$ for optimizing coverage.

- *cost measure* $h_{\text{c}} = c \cdot p - (1-c) \cdot n$

allows to directly trade off consistency and coverage with a parameter c. $c = 0$ only considers consistency, $c = 1$ only coverage. If $c = 1/2$, the resulting heuristic is equivalent to h_{acc}. The isometrics of this heuristics are parallel lines, with a slope of $(1-c)/c$.

- *relative cost measure* $h_{\text{c}_\text{r}} = c_r \cdot h_{\text{tpr}} - (1-c_r) \cdot h_{\text{fpr}}$

trades off the true positive rate and the false positive rate. This heuristic is quite similar to h_{c}. In fact, for any particular dataset, the cost measure and the relative cost measure are equivalent if $c_r = \frac{P}{P+N} \cdot c$. However, the performance of fixed values of c and c_r over a wide variety of datasets with different class distributions will differ. Clearly, setting $c_r = 1/2$ implements h_{WRA}.

- *F-measure* $h_{\text{F}} = \frac{(\beta^2+1) \cdot h_{\text{prec}} \cdot h_{\text{rec}}}{\beta^2 \cdot h_{\text{prec}} + h_{\text{rec}}}$

The F-measure (Salton and McGill 1986) has its origin in Information Retrieval and trades off the basic heuristics h_{prec} and h_{rec}. Its isometrics are illustrated in Fig. 2. Basically, the

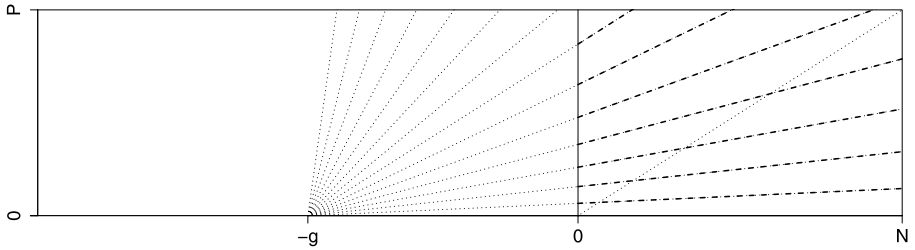


Fig. 2 General behavior of the F-measure

isometrics are identical to those of precision, with the exception that the rotation point does not originate in $(0, 0)$ but in a point $(-g, 0)$, where g depends on the choice of β. If $\beta \to 0$, the origin moves towards $(0, 0)$, and the isometrics correspond to those of h_{prec}. The more the parameter is increased the more the origin of the isometrics is shifted in the direction of the negative N-axis. The observable effect is that the lines in the isometrics becomes flatter and flatter. Conversely if $\beta \to \infty$ the resulting isometrics approach those of h_{rec} which are horizontal parallel lines.

- *m-estimate* $$h_{\text{m}} = \frac{p+m\cdot\frac{P}{P+N}}{p+n+m}$$

The idea of this parametrized heuristic (Cestnik 1990) is to presume that a rule covers m training examples a priori, maintaining the distribution of the examples in the training set ($m \cdot P/(P + N)$ examples are positive). For $m = 2$ and assuming an equal example distribution ($P = N$), we get h_{Lap} as a special case.

If we inspect the isometrics in relation to the different parameter settings, we observe a similar behavior as discussed above for the F-measure, except that now the origin of the turning point does not move on the N-axis, but it is shifted in the direction of the negative diagonal of the coverage space (cf. Fürnkranz and Flach 2005, for an illustration). $m = 0$ corresponds to precision, and for $m \to \infty$ the isometrics become increasingly parallel to the diagonal of the coverage space, i.e., they approach the isometrics of h_{WRA}. Thus, the m-estimate trades off h_{prec} and h_{WRA}.

- *Klösgen* $$h_{\omega} = (h_{\text{cov}})^{\omega} \cdot \left(h_{\text{prec}} - \frac{P}{P+N}\right)$$

trades off *Precision Gain* (the increase in precision compared to the default distribution $P/(P + N)$) and *Coverage*. The isometrics of *Precision Gain* on their own behave like the isometrics of precision, except that their labels differ (the diagonal now always corresponds to a value of 0).

Setting $\omega = 1$ results in WRA, and $\omega = 0$ yields *Precision Gain*. Thus, the Klösgen measure starts with the isometrics of h_{prec} and first evolves into those of h_{WRA}, just like the m-estimate. However, the transformation takes a different route, with non-linear isometrics. The first two graphs of Fig. 3 shows the result for the parameter settings $\omega = 0.5$ and $\omega = 1$ (WRA), which were suggested by Klösgen.

With a further increase of the parameter, the isometrics converge to h_{cov}. The middle left graph shows the parameter setting $\omega = 2$, which was suggested by Wrobel (1997). Contrary to the previous settings, the isometrics now avoid regions of low coverage, because the influence of the (negative) coverage is increased. A further increase of the parameter results in sharper bends of the isometrics. The influence of WRA (the part parallel to the diagonal) vanishes except for very narrow regions around the diagonal, and the isometrics gradually transform into those of coverage.

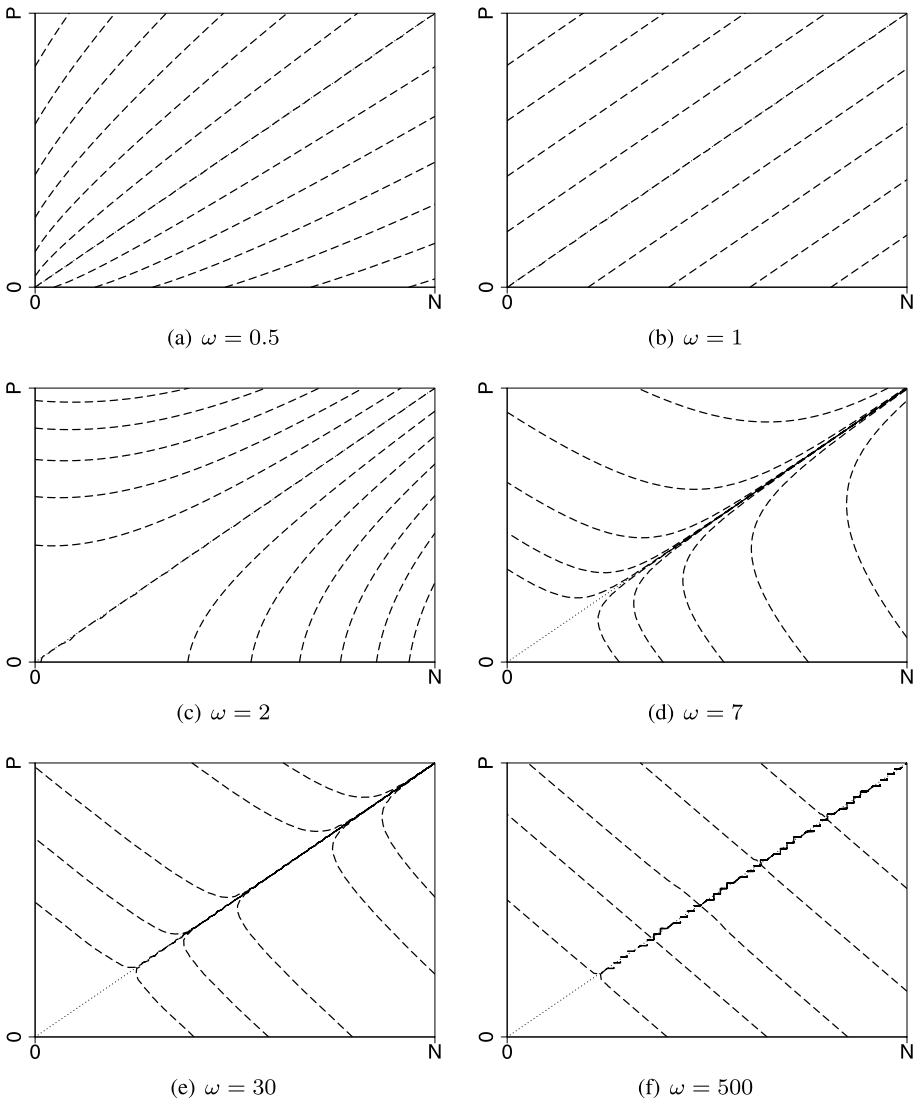


Fig. 3 Klösgen-measure for different settings of ω

Another interesting variation of the Klösgen measure is to divide h_{cov} by $1 - h_{\text{cov}}$ instead of raising it to the ω-th power. It has been shown before Klösgen (1992) that this is equivalent to $h_{\text{correlation}}$. This family of measures was first proposed by Klösgen (1992), and has been frequently used for subgroup discovery.

4 Experimental setup

The primary goal of our experimental work is to determine search heuristics that are optimal in the sense that they will result in the best overall performance on a wide variety of datasets.

Thus, we have to keep several things in mind. First, our results should be valid for a wide variety of datasets with different characteristics. Second, we have to be careful not to overfit the selected datasets. Finally, we have to select ways for assessing the performance of a heuristic. In this section, we will describe our choices for addressing these concerns.

4.1 The datasets

We arbitrarily selected the following 27 *tuning datasets* from the UCI-Repository (Asuncion and Newman 2007).

anneal, audiology, breast-cancer, cleveland-heart-disease, contact-lenses, credit, glass2, glass, hepatitis, horse-colic, hypothyroid, iris, krkp, labor, lymphography, monk1, monk2, monk3, mushroom, sick-euthyroid, soybean, tic-tac-toe, titanic, vote-1, vote, vowel, wine.

Only these datasets were used for making comparative choices between different heuristics (e.g., for optimizing a parameter of a heuristic, or for metalearning a heuristic).

To check the validity of the optimization results, we selected 30 additional *validation datasets*.

auto-mpg, autos, balance-scale, balloons, breast-w, breast-w-d, bridges2, colic, colic.ORIG, credit-a, credit-g, diabetes, echocardiogram, flag, hayes-roth, heart-c, heart-h, heart-statlog, house-votes-84, ionosphere, labor-d, lymph, machine, primary-tumor, promoters, segment, solar-flare, sonar, vehicle, zoo.

These datasets were used for validation only, no choices were based on the results of these datasets.

4.2 Evaluation methods

Our primary method for evaluating heuristics is to use these heuristics inside the rule learner, and observe the resulting predictive accuracies across a variety of datasets. On each individual dataset, predictive accuracy is estimated using a single *stratified 10-fold cross validation*, as implemented in *Weka* (Witten and Frank 2005). As we have a large number of different individual results, a key issue is how to combine the individual results into an overall performance measure.

In the following, we describe the metrics we used, where p_i (n_i) denotes the covered and P_i (N_i) are the total number of positive (negative) examples for the i-th dataset out of a total of m datasets.

Macro-Averaged-Accuracy is the standard average of the accuracies on the m individual datasets.

$$Acc_{\text{macro}} = \frac{1}{m} \sum_{i=1}^{m} \frac{p_i + (N_i - n_i)}{P_i + N_i}$$

A key disadvantage of this method is that the variance of the performances of the algorithms may differ considerably, and the differences in average performance may be dominated by the performance on a few high-variance datasets. Thus, we also consider *Micro-Averaged Accuracy*, which assigns the same weight to each misclassified example. In effect, this method assigns a higher weight to datasets with many examples and those with few examples get a smaller weight.

Micro-Averaged-Accuracy is the fraction of correctly classified examples in *all* examples in the union of all examples of the different datasets.

$$Acc_{\text{micro}} = \frac{\sum_{i=1}^{m}(p_i + N_i - n_i)}{\sum_{i=1}^{m}(P_i + N_i)}$$

As there are large differences in the variances of the accuracies of the individual datasets, one could also focus only on the *ranking* of the heuristics and neglect the magnitude of the accuracy differences. Small random variations in ranking performance will cancel out over multiple datasets, but if there is a consistent small advantage of one heuristic over the other this will be reflected in a substantial difference in the average rank.

Average Rank is the average of the individual ranks r_i on each dataset. All heuristics were ranked after their macro-accuracy on each dataset. For heuristics that got an equal accuracy the rank was computed by averaging their individual ranks. For example, if four heuristics share rank 2, 3, 4 and 5 the rank for each of them would be 3.5.

$$Rank = \frac{1}{m}\sum_{i=1}^{m} r_i$$

In addition, we also measured the *Size* of the learned theories by the average number of conditions.

Average Size is the average number of conditions of the rule sets R_i.

$$Size = \frac{1}{m}\sum_{i=1}^{m} |R_i|$$

As mentioned above, we used 27 sets for finding the optimal parameters, and 30 additional sets for checking the validity of the found values. In order to assess this validity, we compute the *Spearman Rank Correlation* between the rankings of the various heuristics on these two sets (different parameterizations of the same heuristic are counted as separate heuristics).

Spearman Rank Correlation Given two (averaged and rounded) rankings r_i and r_i' for the heuristics $h_i, i = 1 \ldots k$, the *Spearman Rank Correlation* ρ is defined as

$$\rho = 1 - \frac{6}{m \cdot (m^2 - 1)}\sum_{i=1}^{k}(r_i - r_i')^2$$

In the metalearning experiments, we will train a function to predict the heuristic values on a separate test set (as opposed to those that can be directly measured on the training set). We evaluated the fit of this learned heuristic function to the target values in terms of its *mean absolute error*, again estimated by one iteration of a 10-fold cross validation on each individual training set.

Mean Absolute Error is the deviation of the predicted heuristic value h' from the true target value h, averaged over all n instances (the union of all instances in the test folds of the cross-validation)

$$MAE(h') = \frac{1}{n}\sum_{j=0}^{n} |h'(j) - h(j)|$$

Note, however, that the mean absolute error measures the error made by the regression model on unseen data. A low mean absolute error on a dataset does not implicate that the function works well as a heuristic. For example, a systematic, large over-estimation of the heuristic value may result in a higher absolute error than a small random fluctuation around the correct value, but may produce a much better performance if the correct ordering of values is preserved.

5 Optimization of parametrized heuristics

In this section, we will determine optimal parameters for the five parametrized rule evaluation metrics that we introduced in Sect. 3.4. We will analyze the average accuracy of the different heuristics under various parameter settings, identify optimal parameter settings, compare their coverage space isometrics, and evaluate their general validity.

5.1 Search strategy

This section describes our method for searching for the optimal parameter setting. Our expectation was that for all heuristics, a plot of accuracy over the parameter value will roughly result in an inverse U-shape, i.e., there will be overfitting for small parameter values and over-generalization for large parameter values, with a region of optimality in between.

Thus, we adopted a greedy search algorithm that continuously narrows down the region of interest. First, it tests a wide range of intuitively appealing parameter settings to get an idea of the general behavior of each of the five parametrized heuristics. The promising parameters were further narrowed down until we had a single point that represents a region of optimal performance.

Algorithm 3 shows the algorithm in detail. We start with a lower (a) and upper (b) bound of the region of interest, and sample the space between them with a certain interval width i. For each sampled parameter value, we estimate its macro-averaged accuracy on all tuning datasets, and, based on the obtained results, narrow down the values a, b, and i.

Algorithm 3 SEARCHBESTPARAMETER($a, b, i, h, dataSets$)

```
# global parameter
acc_former ← acc_best
# initialize candidate params
params ← CREATELIST(a, b, i)
p_best ← GETBESTPARAM(h, params, dataSets)
acc_best ← GETACCURACY(p_best)
# stop if no substantial improvement (t = 0.001)
if (acc_best − acc_former) < t then
  return p_best
end if
# continue the search with a finer resolution
SEARCHBESTPARAMETER(p_best − i/2, p_best + i/2, i/10, h, dataSets)
```

Table 1 A sample parameter search

Run	Set which has to be searched	Increment	Best parameter	Accuracy
1	$\{0.1, \ldots, 1.0\}$	0.1	0.4	84.5658
2	$\{0.35, \ldots, 0.45\}$	0.01	0.42	84.6852
3	$\{0.415, \ldots, 0.425\}$	0.001	0.418	84.7015
4	$\{0.4175, \ldots, 0.4185\}$	0.0001	0.4176	84.7045
5	$\{0.41755, \ldots, 0.41765\}$	0.00001	0.4176	84.7045

Intuitively, the farther the lower border a and the upper border b of the interval are away from the best parameter p_{best}, and the denser the increment, the better are our chances to find the optimal parameter, but the higher are the computational demands. As a compromise, we used the following approach for adjusting the values of these parameters:

$$a \leftarrow p_{\text{best}} - \frac{i}{2}, \qquad b \leftarrow p_{\text{best}} + \frac{i}{2} \quad \text{and} \quad i \leftarrow \frac{i}{10}$$

This procedure is repeated until the accuracy does not increase significantly. As we compare macro-averaged accuracy values over several datasets, we adopted a simple approach that stops whenever the accuracy improvement falls below a threshold $t = 0.001$.

For illustration, Table 1 shows a sample search.

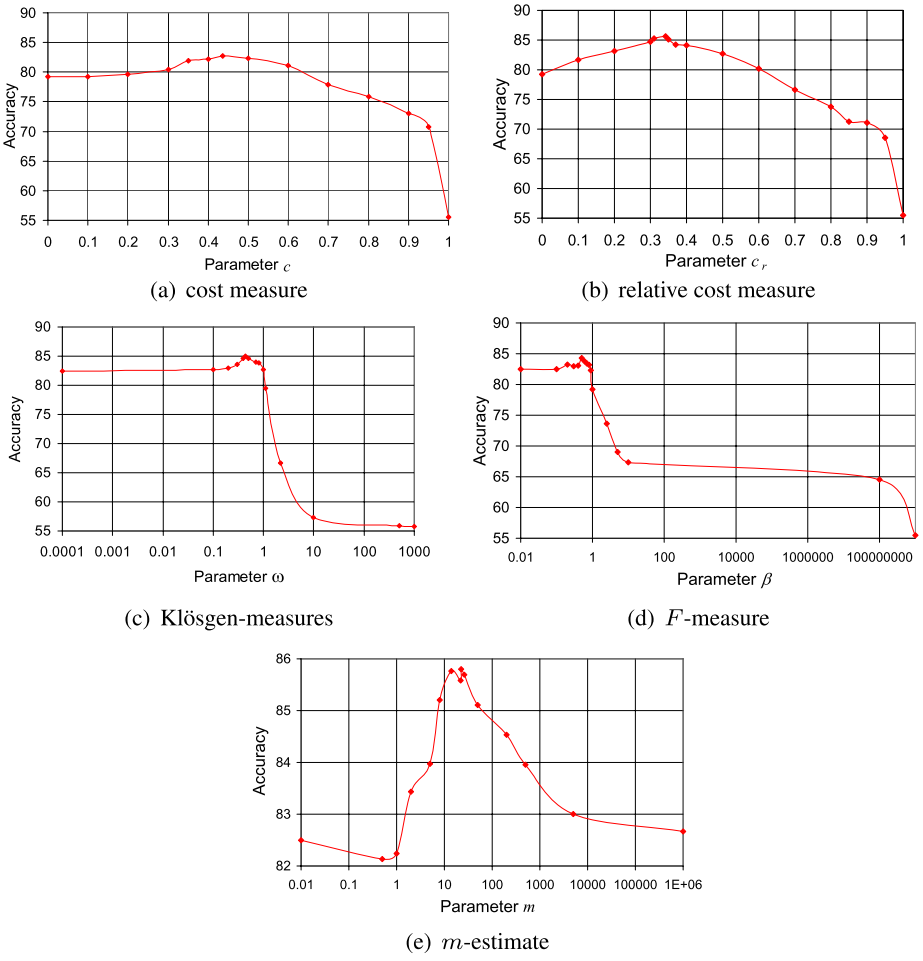
Obviously, the procedure is greedy and not guaranteed to find a global optimum. In particular, there is a risk to miss the best parameter due to the fact that the global best parameter may lie under or above the borders (if the best one so far is 1 for example, the interval that would be searched is [0.5, 1.5]; if the global optimum is 0.4, it would not be detected). Furthermore, we may miss a global optimum if it hides between two apparently lower values. If the curve is smooth, these assumptions are justified, but on real-world data we should not count on this.

The second point means that the procedure may miss a global optimum by only refining one candidate parameter at a time and may therefore get stuck in a local optimum. This is a typical problem of hill climbing search algorithms. As a remedy the best n parameters can be refined simultaneously. This is also known as beam search which is often used to avoid situations where the search get stuck in local optima. To make a good choice for the number of parameters that are kept in the beam is not trivial. Due to this the number of candidate parameters is limited to 3 (all experiments confirmed that this is sufficient). The first problem could be addressed by re-searching the entire interval at a finer resolution, but, for the sake of efficiency, we chose the simpler version.

However, also note that it is not really important to find an absolute global optimum. If we can identify a region that is likely to contain the best parameter for a wide variety of datasets, this would already be sufficient for our purposes. We interpret the found values as good representatives for optimal regions.

5.2 Optimal parameters for the five heuristics

Our first goal was to obtain optimal parameter settings for the five heuristics. As discussed above, the found values are not meant to be interpreted as global optima, but as representatives for regions of optimal performance. Figure 4 shows the obtained performance curves. Note that the parameters for Klösgen, F-measure and m-estimate are plotted on a logarithmic scale.



(a) cost measure (b) relative cost measure

(c) Klösgen-measures (d) F-measure

(e) m-estimate

Fig. 4 Macro-averaged accuracy over parameter values for the five parametrized heuristics

5.2.1 Cost measures

Figures 4 (a) and (b) show the results for the two cost measures. Compared to the other measures, these curves are comparably smooth, and optimal values could be identified quite easily. Optimizing only the consistency (i.e., minimizing the number of negative examples without paying attention to the number of covered positives) has a performance of close to 80%. Not surprisingly, this can be improved considerably for increasing values of the parameters c and c_r. The best performing values were found at $c = 0.437$ (for the cost metric) and $c_r = 0.342$ (for the relative cost metric). Further increasing these values will decrease performance because of over-generalization. If the parameter approaches 1, there is a steep descent because optimizing only the number of covered examples without regard to the covered negatives is, on its own, a very bad strategy.

It is interesting to interpret the found values. For the cost metric, the optimal value $c = 0.437$ corresponds to a slope of $(1 - c)/c \approx 1.3$, i.e., one false positive corresponds

to approximately 1.3 true positives. Thus, consistency is favored over coverage. More interestingly, this bias towards consistency not only holds for absolute numbers but also for the true positive and false positives *rates*. Note that weighted relative accuracy, which has been previously advocated as rule learning heuristic (Todorovski et al. 2000), corresponds to a value of $c_r = 0.5$, equally weighting false positive rate and true positives rate. Comparing this to the optimal region for this parameter, which is approximately between 0.3 and 0.35, it can be clearly seen that it pays off to give a higher weight to the false positive rate, thereby favoring consistency over coverage.[1]

It is also interesting to compare the results of the absolute and relative cost measures: although, as we have stated above, the two are equivalent in the sense that for each individual dataset, one can be transformed into each other by picking an appropriate cost factor, the relative cost measure has a clearly better peak performance exceeding 85%. Thus, it seems to be quite important to incorporate the class distribution $P/(P + N)$ into the evaluation metric. This is also confirmed by the results of the m-estimate and the Klösgen measures.

5.2.2 Klösgen measures

Figure 4 (c) shows the results for the Klösgen measures. In the region from 0.1 to 0.4 the accuracy increases continuously until it reaches a global optimum at 0.4323, which achieves an average accuracy of almost 85%. After the second iteration of the SEARCHBESTPARAMETER algorithm, no better candidate parameters than 0.4 were found. The accuracy decreases again with parameterizations greater than 0.6. As illustrated in Fig. 3, the interval [0, 1] describes the trade-off between *Precision* ($\omega = 0$) and WRA ($\omega = 1$), whereas values of $\omega > 1$ trade off between WRA and *Coverage*. The bad performance in the region of $\omega > 1$ (presumably due to over-generalization) surprised us, because we originally expected that the good performance known from subgroup discovery (Wrobel 1997) might carry over to classification rule induction. The main problem here seems that as long as a certain level of positive coverage is guaranteed the negative coverage can be increased without negative effect. This becomes even clearer in the plot of Fig. 3 (d) for $\omega = 7$ were the positive and negative coverage are weighted nearly equally. Finally, as the isometrics evolve to those of h_{cov} the difference of the positive and negative coverage vanishes completely.

5.2.3 F-measure

For the F-measure the same interval as with the Klösgen measures is of special interest (Fig. 4 (d)). Already after the first iteration, the parameter 0.5 turned out to have the highest accuracy of 82.2904%. A better one could not be found during the following iterations. After the second pass two other candidate parameters, namely 0.493 with 84.1025% and 0.509 with 84.2606% were found. But both of them could not be refined to achieve a higher accuracy and were therefore ignored. The main difference between the Klösgen measures and the F-measure is that for the latter, the accuracy has a steep descent at a very high parametrization of $1 \cdot E^9$. At this point it overgeneralizes in the same way as the Klösgen measures or the cost measures (at about 55%).

[1] Interestingly, the optimal value of $c = 0.342$ corresponds almost exactly to the micro-averaged default accuracy of the largest class (for both tuning and validation datasets). We are still investigating whether this is coincidental or not.

5.2.4 *m*-estimate

The behavior of the m-estimate differs from the other parametrized heuristics in several ways. In particular, it proved to be more difficult to search. For example, we can observe a small descent for low parameter settings (Fig. 4 (e)). The main problem was that the first iteration exhibited no clear tendencies, so the region in which the best parameter should be could not be restricted. As a consequence, we re-searched the interval $[0, 35]$ with a smaller increment of 1 because all parameters greater than 35 got accuracies under 85.3% and we had to restrict the area of interest. After this second iteration there were 3 candidate parameters, from which 14 achieves the greatest accuracy. After a second run, 23.5 became optimal, which illustrates that it was necessary to maintain a list of candidate parameters. After a few more iterations, we found the optimal parameter at 22.466. The achieved accuracy of 85.87% was the optimum among all heuristics.

5.3 Experimental results of the tuned heuristics

In this section, we compare the parameters which have been found for the five heuristics (cf. also Table 2). Then we show experiments to make sure that our results are not only due to overfitting of the 27 tuning datasets. We will then also describe experiments in which the tuned heuristics are used in two other rule learning algorithms, which have not been implemented by us, namely different versions of CN2 and Ripper.

5.3.1 *Results on the 27 tuning datasets*

Table 2 shows the results of the different heuristics on the 27 datasets, on which the parameters were tuned. We show micro- and macro-averaged accuracy, the average rank of the method on the datasets, and the average size of the learned rule sets. The numbers in brackets indicate the ranking of the methods according to each method. The table is sorted according to macro-averaged accuracy.

According to this metric, the m-estimate and the relative cost measure clearly outperformed the other parametrized heuristics, as well as the standard heuristics, which we have also briefly described in Sect. 3.4. Interestingly, the relative cost measure performs much worse with respect to micro-averaged accuracy, indicating that it performs rather well on small datasets, but worse on larger datasets. These two heuristics also outperform JRip (the *Weka*-implementation of Ripper; Cohen 1995) on these datasets.

Interestingly the cost metric performed rather bad. We think that this is due to the fact that this is the only parametrized heuristic that does not include information about the class distribution into its evaluation function. The m-estimate, the Klösgen measures, and the relative cost metric directly include the a priori probability of the positive class ($P/(P+N)$), whereas the F-measure only normalizes the positive examples. The results from our metalearning experiments (Sect. 6) will support this hypothesis.

In terms of theory size, *weighted relative accuracy* is the clear winner, with a predictive performance that exceeds the one of most other standard heuristics. This confirms the results of Todorovski et al. (2000). However, there is a large gap to the performance of JRip and the parametrized heuristics. This indicates that, while Precision and Laplace obviously overfit the data, WRA has a tendency to over-generalize.

Obviously, the good results of the parametrized heuristics must be put into perspective because the parameters of the heuristics were optimized to perform well on this subset of datasets (they were, however, not optimized on individual datasets). Thus, in order to get a fair comparison, it seems necessary to evaluate the methods on independent datasets, which were not used for tuning the parameters.

Table 2 Results of the optimal parameter settings (identified by their parameters), other commonly used rule learning heuristics, and JRip (Ripper) with and without pruning on the 27 tuning datasets, sorted by their macro-averaged accuracy

Heuristic	Average accuracy		Average	Average
	Macro	Micro	rank	size
m-estimate ($m = 22.466$)	85.87 (1)	93.87 (1)	4.54 (1)	36.85 (4)
Relative cost ($c_r = 0.342$)	85.61 (2)	92.50 (6)	5.54 (4)	26.11 (3)
Klösgen ($\omega = 0.4323$)	84.82 (3)	93.62 (3)	5.28 (3)	48.26 (8)
JRip	84.45 (4)	93.80 (2)	5.12 (2)	16.93 (2)
F-measure ($\beta = 0.5$)	84.14 (5)	92.94 (5)	5.72 (5)	41.78 (6)
JRip-P	83.88 (6)	93.55 (4)	6.28 (6)	45.52 (7)
Correlation	83.68 (7)	92.39 (7)	7.17 (7)	37.48 (5)
WRA	82.87 (8)	90.43 (12)	7.80 (10)	14.22 (1)
Cost measure ($c = 0.437$)	82.60 (9)	91.09 (11)	7.30 (8)	106.30 (12)
Precision	82.36 (10)	92.21 (9)	7.80 (10)	101.63 (11)
Laplace	82.28 (11)	92.26 (8)	7.31 (9)	91.81 (10)
Accuracy	82.24 (12)	91.31 (10)	8.11 (12)	85.93 (9)

Table 3 Results of the optimal parameter settings (identified by their parameters), other commonly used rule learning heuristics, and JRip (Ripper) with and without pruning on the 30 validation datasets, sorted by their macro-averaged accuracy

Heuristic	Average accuracy		Average	Average
	Macro	Micro	rank	size
JRip	78.98 (1)	82.42 (1)	4.72 (1)	12.20 (2)
Relative cost ($c_r = 0.342$)	78.87 (2)	81.80 (3)	5.28 (3)	25.30 (3)
m-estimate ($m = 22.466$)	78.67 (3)	81.72 (4)	4.88 (2)	46.33 (4)
JRip-P	78.54 (4)	82.04 (2)	5.38 (4)	49.80 (6)
Klösgen ($\omega = 0.4323$)	78.46 (5)	81.33 (6)	5.67 (6)	61.83 (8)
F-measure ($\beta = 0.5$)	78.12 (6)	81.52 (5)	5.43 (5)	51.57 (7)
Correlation	77.55 (7)	80.91 (7)	7.23 (8)	47.33 (5)
Laplace	76.87 (8)	79.76 (8)	7.08 (7)	117.00 (10)
Precision	76.22 (9)	79.53 (9)	7.83 (10)	128.37 (12)
Cost measure ($c = 0.437$)	76.11 (10)	78.93 (11)	8.15 (11)	122.87 (11)
WRA	75.82 (11)	79.35 (10)	7.82 (9)	12.00 (1)
Accuracy	75.65 (12)	78.47 (12)	8.52 (12)	99.13 (9)

5.3.2 Validity of the results on 30 validation datasets

In order to make sure that our results are not only due to overfitting of the 27 tuning datasets, we also evaluated the found parameter values on 30 new validation datasets. The results are summarized in Table 3. The numbers in brackets describes the rank of each heuristic according to the measure of the respective column.

Qualitatively, we can see that the relative performance of the heuristics in comparison to each other, and in comparison to the standard heuristics does not change much. The only

Table 4 Spearman rank correlation between rankings of Table 2 and of Table 3

Heuristic	Average accuracy		Average	
	Macro	Micro	Rank	Size
Spearman	0.85315	0.92308	0.88112	0.98601

Table 5 Win/loss/tie statistics and the p-values of the sign test for the macro-averaged accuracy of the optimized heuristics vs. standard heuristics on the 30 validation datasets

Win/loss/tie p-value	Precision	Laplace	Accuracy	WRA	Corr.	Sum
Cost	12/17/1 *0.458*	11/17/2 *0.345*	13/16/1 *0.711*	15/14/1 *1.000*	13/14/3 *1.000*	64/78/8
Relative cost	18/9/3 *0.122*	18/8/4 *0.0755*	23/7/0 *0.00522*	20/6/4 *0.00936*	19/9/2 *0.0872*	98/39/13
m-estimate	24/6/0 *0.00143*	20/9/1 *0.0614*	19/10/1 *0.136*	19/10/1 *0.136*	20/6/4 *0.00936*	102/41/7
Klösgen	22/8/0 *0.161*	18/10/2 *0.185*	23/7/0 *0.00522*	19/10/1 *0.136*	18/8/4 *0.0755*	100/43/7
F-measure	21/6/3 *0.00592*	18/11/1 *0.265*	24/4/2 *0.00018*	21/9/0 *0.0428*	17/9/4 *0.169*	101/39/10
Sum	97/46/7	85/55/10	102/44/4	94/49/7	87/46/17	

obvious difference is the considerably better performance of JRip, which indicates that some amount of overfitting has happened in the optimization phase. However, the performance of the best metrics is still comparable to the performance of JRip, although the latter achieves this performance with much smaller rule sizes.

Table 4 shows the Spearman rank correlation coefficients between the ranking of the heuristics on the tuning datasets and on the validation datasets. For all four measurements, we observe a correlation >0.85, which makes us confident that the found optimal parameters are not overfitting the tuning datasets, but will also work well on new datasets.

Table 5 gives a more fine-grained view on the performances of the optimized heuristics versus the standard heuristics on the 30 validation datasets. It shows for each pair of optimized and standard heuristic the number of wins, losses, and ties for the optimized heuristic. Below these three values, we show the p-value for a sign test with these values (i.e., the error probability for rejecting the hypothesis that the two heuristics are equal). The last column shows the sum of the values of the previous columns, i.e., they show how often the heuristic in this row has outperformed any of the heuristics in the columns. The row sums in the last row can be interpreted accordingly.

Again, we can see that, with the exception of the cost metric, all optimized heuristics outperform the standard heuristics on the majority of the datasets. There is not a single case where a standard heuristic has more wins than an optimized heuristic. In fact, each optimized heuristics has at least 17 wins and not more than 10 losses. In many cases, the margin is much larger, and many of the differences are highly significant, even with the crude sign test.

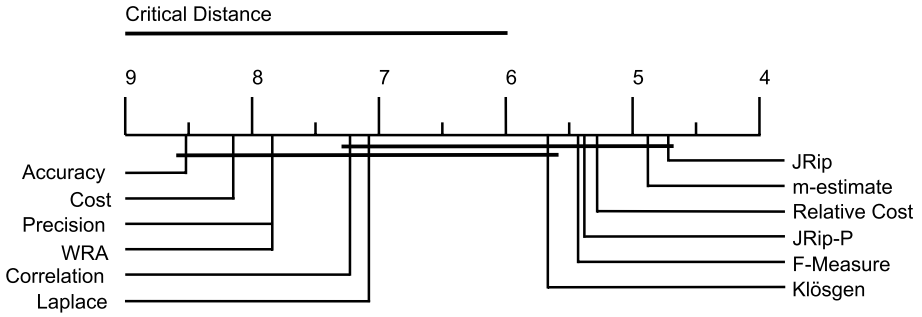


Fig. 5 Comparison of all heuristics against each other with the Nemenyi test. Groups of heuristics that are not significantly different (at $p = 0.05$) are connected

Table 6 Macro-averaged accuracy of the five heuristics when used in JRip (with and without pruning) and in CN2 (in unordered and decision-list mode)

Heuristic	JRip	JRip-P	CN2-u	CN2-dl	SeCo
Default heuristic	79.19	79.01	78.06	78.44	–
m-estimate	78.64	78.36	77.86	78.17	79.22
Klösgen	78.04	78.10	78.42	78.70	78.71
F-measure	78.54	77.13	77.62	77.98	78.70
Relative cost	77.71	78.03	77.53	77.80	79.09
Cost	74.57	70.54	72.97	75.75	76.89

Finally, Fig. 5 displays a comparison of the ten heuristics and the two versions of JRip done with the Nemenyi test as suggested by Demsar (2006). The results from above are verified, which means that only the Klösgen measures and the cost metric are not significantly better than Accuracy, Precision and WRA. All other heuristics including JRip outperform these heuristics significantly. Furthermore, even though correlation and Laplace are not significantly worse, we notice a large gap between all standard heuristics and the tuned ones (except the cost metric).

5.3.3 Validity of the results with other algorithms

We also implemented the heuristics in the original implementation of CN2[2] and JRip, the *Weka*-Implementation of Ripper. Table 6 displays the results for evaluating these algorithms on the 30 validation data sets. 2 datasets were left out because they contain missing class values that cannot be handled by CN2. Therefore, all results displayed in Table 6 are calculated on the remaining 28 datasets. JRip was used with and without pruning (-P). The other parameters were left at default values. In particular, CN2 was run in ordered and unordered mode and with the default beam width of 5.

Table 6 summarizes the results of these experiments. With respect to the relative order of the parametrized metrics, they essentially confirm our previous results: the *m*-estimate,

[2]Available from http://www.cs.utexas.edu/users/pclark/software/.

the relative cost measure, the Klösgen measure and the F-measure are essentially indistinguishable (with a slight over-all advantage for the m-estimate), with the cost measure clearly lagging behind.

Compared to the original implementations, however, the results are not entirely as expected. First, we can note that JRip's gain heuristic seems to outperform our heuristics. This is not implausible, because we have already observed above that heuristics that take the prior class distribution into account outperform heuristics that don't. Gain heuristics can be interpreted as normalizing the example distribution so that the distribution of the covered examples of the previous rule is used as a prior distribution for finding the next literal (Fürnkranz and Flach 2004). While, for reasons outlined in Sect. 2.2, it is beyond the scope of this paper, the question whether gain heuristics are generally preferable to absolute heuristics certainly deserves further investigation. We are currently working on this.

More surprising to us, however, was that our metrics did not improve CN2's default heuristic (Laplace) in terms of predictive accuracy although they did learn simpler rules in many cases. As we have discussed in Sect. 2.2, we consider the differences between our SeCo implementation and CN2's original implementation to be only minor, the only major difference being the strategy for handling multiple classes. Nevertheless, the improvements over Laplace, which we have observed in our implementation, do not seem to carry over to this implementation. Apparently, the differences between the algorithms are larger than we had expected, which is also witnessed by the large difference in predictive accuracy between CN2 and SeCo.

In general, while the learned values are certainly reasonable for other algorithms, our measures performed the best inside our own algorithm. Our SeCo implementation outperforms both CN2 and JRip when used with our learned metrics. So, while the good performance seems to carry over to new datasets, the metrics seem to capture some aspects that are specific to the algorithm used. We will discuss this issue in a bit more detail in Sect. 7. In some sense, these results correspond to some more recent results (Janssen and Fürnkranz 2009), where we showed that different heuristics may exhibit very different behaviors when used with different beam widths.[3]

5.4 Interpretation of the learned heuristics

Figure 6 shows the isometrics of the best parameter settings of the m-estimate, the F-measure, the Klösgen-measure, and the relative cost measure. It is interesting to compare the implemented preference structures. The Klösgen measure and the m-estimate appear to implement quite similar behavior. Their isometrics have almost the same shape, except that those of the Klösgen measures are slightly non-linear. The F-measure is also quite similar in the upper left region (high coverage and high consistency), but differs slightly in the low coverage regions, where it is necessarily parallel to the N-axis. The isometrics for the relative cost measure are confined to parallel lines. The slope of these isometrics seem to form an average: in high coverage and high consistency regions the slope is less steep than in the other heuristics, while in low coverage and low consistency regions it is considerably steeper. In any case, the slope is steeper than the diagonal, i.e., it is obvious that this heuristic gives a higher weight to consistency than to coverage.

[3]Note that the results described here are also from different beam widths (CN2 has a default beam width of 5, while SeCo uses hill-climbing). We had also tried to run CN2 with hill-climbing (beam size 1), and the results were qualitatively similar. However, we are not sure whether hill-climbing works correctly in CN2. In particular, we noticed that the results were considerably worse because numerical attributes are practically ignored in that mode. Thus, we decided to omit these results from the paper.

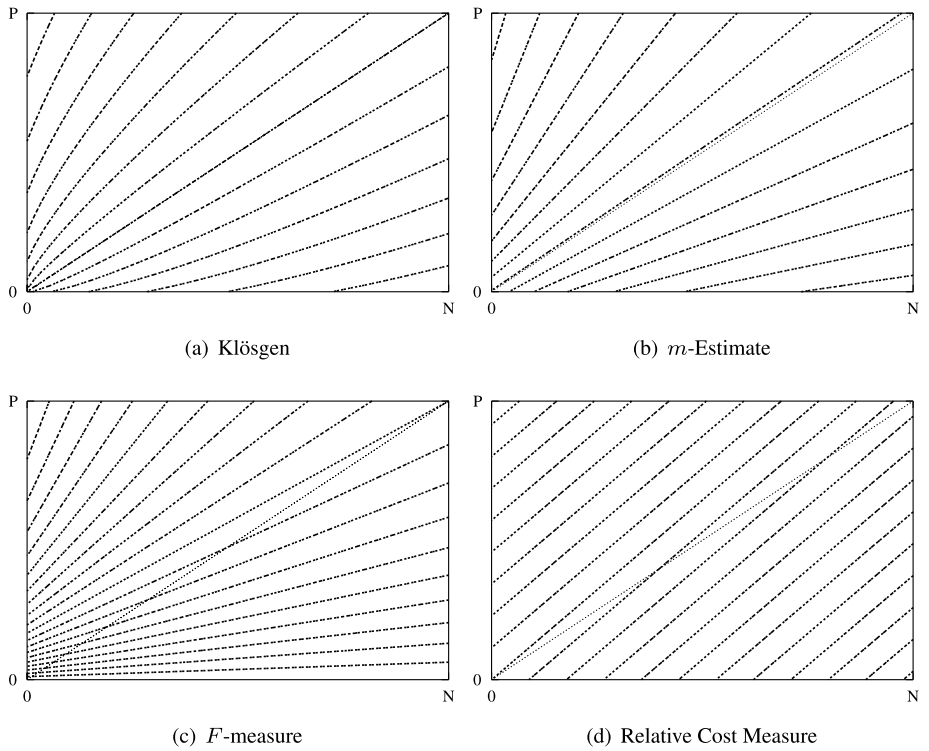


Fig. 6 Isometrics of the best parameter settings

6 Metalearning of rule learning heuristics

While the previous section has focused on determining optimal parameters for a given functional form, we will now try to learn a function $h(p, n)$ from scratch. Thus, in this part of the paper we do not optimize parameters any more but we try to take a different route. First it has to be defined how a function without a predefined form can be learned. In the following, we will therefore frame this problem as a metalearning task, in which we try to predict the "true" performance of a rule on the test set.

6.1 Metalearning scenario

The key issue for our work is how to define the metalearning problem. It is helpful to view the rule learning process as a reinforcement learning problem: Each (incomplete) rule is a state, and all possible refinements (e.g., all possible conditions that can be added to the rule) are the actions. The rule-learning agent repeatedly has to pick one of the possible refinements according to their expected utility until it has completely learned the rule. Then, the learner receives a reinforcement signal (e.g., the estimated accuracy of the learned rule), which can then be used to adjust the utility function. After a (presumably large) number of learning episodes, the utility function should converge to a heuristic that evaluates a candidate rule with the quality of the *best* rule that can be obtained by refining the candidate rule.

However, for practical purposes this scenario appears to be too complex. Burges (2006) has tried a reinforcement learning approach on this problem, but with disappointing results.

For this reason, we tried another, conceptually simpler approach, which tries to learn the same function in a supervised fashion: Each rule is evaluated on a separate test set, in order to get an estimate of its true performance. As a target value, we can either directly use the candidate rule's performance (*immediate reward*), or we can use the performance of its best refinement (*delayed reward*). We evaluated both approaches.

6.1.1 Meta data generation

We have noted above, that heuristics typically depend on the number of true and false positives, and on the total number of positive and negative examples. However, most heuristics model non-linear dependencies between these values. In order to make the task for the learner easier, we will not only characterize a rule by the values p, n, P, and N, but in addition also use the following parameters as input for the metalearning phase:

- $tpr = \frac{p}{P}$, the true positive rate of the rule
- $fpr = \frac{n}{N}$, the false positive rate of the rule
- $Prior = \frac{P}{P+N}$, the a priori distribution of positive and negative examples
- $prec = \frac{p}{p+n}$, the fraction of positive examples covered by the rule

Thus, we characterize a rule r by an 8-tuple

$$h(r) \leftarrow h(P, N, Prior, p, n, tpr, fpr, prec)$$

In Sect. 6.2.2, we will also consider the *rule length* l as an additional input.

As explained above, we try to model the relation of the rule's statistics measured on the training set and its "true" performance, which is estimated on an independent test set. Thus, a meta-training instance consists of the abovementioned characteristics for the corresponding rule. The training signal is the performance of the rule on the test set. For assessing the performance of the rule, we typically use its out-of-sample precision, but, again, we have also experimented with other choices.

As we want to guide the entire rule learning process, we need to record this information not only for final rules—those that would be used in the final theory—but also for all their predecessors. Therefore all candidate rules which are created during the refinement process are included in the meta data as well. Algorithm 4 shows this process in detail.

It should be noted, that we ignored all rules that do not cover any instance on the test data. Our reasons for this were that on the one hand we did not have any training information for this rule (the test precision that we try to model is undefined for these rules), and that on the other hand such rules do not do any harm (they won't have an impact on test set accuracy as they do not classify any example).

To ensure that we obtain a set of rules with varying characteristics, the following parameters were modified:

Datasets: All models were trained on the 27 tuning datasets defined in Sect. 4.1.

5×2 *Cross-validation*: For each dataset, we performed 5 iterations of a 2-fold cross-validation. 2-fold cross-validation was chosen because in this case the training and test sets have equal size, so that we don't have to account for statistical variance in the precision or coverage estimates. We performed five iterations with different random seeds. Note that our primary interest was to obtain a lot of rules which characterize the connection between training set statistics and the test set precision. Therefore, we collected statistics for all rules of all folds.

Algorithm 4 GENERATEMETADATA*(TrainSet,TestSet)*

```
# loop until all positive examples are covered
  while POSITIVE(TrainSet) ≠ ∅
    # find the best rule
    Rule ← GREEDYTOPDOWN(TrainSet)
    # stop if it doesn't cover more positives than negatives
    if |COVERED(Rule, POSITIVE(Examples))|
        ≤ |COVERED(Rule, NEGATIVE(Examples))|
        break
    # loop through all predecessors
    Pred ← Rule
    repeat
      # record the training and test coverage
      p ← |COVERED(Pred,POSITIVE(TrainSet))|
      n ← |COVERED(Pred,NEGATIVE(TrainSet))|
      P ← |COVERED(Pred,TOTALPOSTIVE(TrainSet))|
      N ← |COVERED(Pred,TOTALNEGATIVE(TrainSet))|
      l ←LENGTH(Rule)
      p̂ ← |COVERED(Pred,POSITIVE(TestSet))|
      n̂ ← |COVERED(Pred,NEGATIVE(TestSet))|
      # print out meta training instance
      print P, N, P/(P+N), p, n, p/P, n/N, p/(p+n), l
      # print out meta target information
      print p̂, n̂, p̂/(p̂+n̂)
      Pred ← REMOVELASTCONDITION(Pred)
    until Pred = null
    # remove covered training and test examples
    TrainSet ← TrainSet \ COVERED(Rule,TrainSet)
    TestSet ← TestSet \ COVERED(Rule,TestSet)
```

Classes: For each dataset and each fold, we generated one dataset for each class, treating this class as positive and the union of all the others as the negative class. Rules were learned for each of the resulting two-class datasets.

Heuristics: We ran the rule learner several times on the 2-class datasets, each time using a different search heuristic. We used all basic heuristics described in Section 3. As discussed there, these heuristics represent a large variety of learning biases, some overfitting, some overgeneralizing.

In total, our meta dataset contains 87,380 examples.

6.1.2 Metalearning algorithms

We used two different methods for learning functions on the meta data. First, we used a simple *linear regression* using the Akaike criterion (Akaike 1974) for model selection. A key advantage of this method is that we obtain a simple, easily comprehensible form of the

learned heuristic function. Note that the learned function is nevertheless non-linear in the basic dimensions p and n because of the abovementioned non-linear terms that are used as basic features.

Nevertheless, the type of functions that can be learned with linear regression is quite restricted. In order to be able to address a wider class of functions, we also tried a *multilayer perceptron* with back propagation algorithm and sigmoid nodes. We used various sizes of the hidden layer (1, 5, and 10), and trained for one epoch (i.e., we went through the training data once). We have also tried to train the networks with a larger number of epochs, but the results no longer improved. We used the abovementioned numbers of nodes in the hidden layer because we wanted to have a very simple model that can be trained very fast. Nevertheless, the functions that can be learned with one node in the hidden layer are restricted to linear ones. For this reason we also tried 5 and 10 nodes to make sure that we have not selected a model that is too simple to perform reasonable on the metalearning task.

Both algorithms are provided by *Weka* (Witten and Frank 2005) and were initialized with standard parameters. We had also tried a support vector machine for Regression. As the Regression SVMs of *Weka* (SVMReg and SMOReg) could not be trained on the metadata because the datasets were too big, we resorted to the use of LibSVM (Fan et al. 2005). However, the results were comparable to those of the neural network, which in turn was worse than the linear regression. Hence we do not include experimental results of the SVM in the paper.

6.2 Experimental results

In this section, we discuss our experimental results with the metalearning approach. We will start with a straight-forward baseline experiment that uses the meta-data as described in Sect. 6.1.1, and then try to experimentally answer the questions whether inclusion of the rule length improves the result, whether learning in the delayed reward scenario is better than learning from immediate rewards, and whether other heuristic functions perform better than (predicted) precision.

6.2.1 Baseline experiment

In a first experiment, we wanted to see how accurately we can predict the out-of-sample precision of a rule using the meta data as described in Sect. 6.1.1. We trained a linear regression model and a neural network on the eight measurements that we use for characterizing a rule (cf. Sect. 3) using the precision values measured on the test sets as a target function. Table 7 displays results for the linear regression and three neural networks with different numbers of nodes in the hidden layer on the same 30 validation datasets that were used before (cf., Sect. 4). The performances of the three algorithms are quite comparable, with the possible exception of the neural network with 5 nodes in the hidden layer. The heuristic learned by this network induced very large theories (over 1000 conditions on average), and also had a somewhat worse performance in predictive accuracy. In general, the experiments seem to show that a linear combination of the available features is sufficient, and that more nodes in the hidden layer will not yield performance improvements. It can also be seen that, as discussed in Sect. 4.2, a low mean absolute error does not necessarily imply a heuristic that is able to order the rules by their predictive accuracy and therefore works well as rule evaluation measure.

If we compare these results to those of Table 3 (column macro-averaged accuracy), we can see that the learned heuristics outperform all standard heuristics with the exception of correlation. However, they do not quite reach the performance of the optimized parametrized heuristics.

Table 7 Macro- and micro-averaged prediction errors of SECO for several metal-learned heuristics. The second column shows the mean absolute error (MAE) of a cross-validation of the meta-learning training set

Heuristic	MAE	Average accuracy		# conditions
		Macro	Micro	
Linear regression	0.22	77.43%	80.19%	117.6
MLP (1 node)	0.28	77.81%	81.43%	121.3
MLP (5 nodes)	0.27	77.37%	80.45%	1085.8
MLP (10 nodes)	0.27	77.53%	80.27%	112.7

6.2.2 Significance of rule length

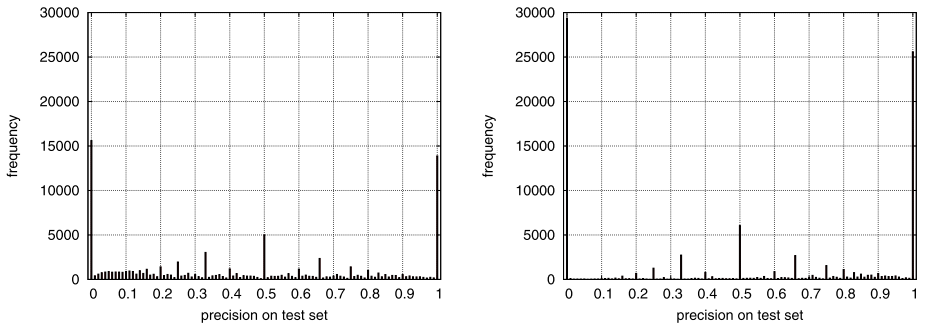
Some rule learning algorithms include the length of the learned rule into their evaluation function. For example, the ILP algorithm Progol (Muggleton 1995) uses $p - n - l$ as a search heuristic for a best-first search. The first part, $p - n$, directly optimizes accuracy (for a fixed dataset, i.e., where the total number of positive (P) and negative (N) examples are fixed), and the length of the rule is used to add an additional bias for simpler rules. However, as longer rules typically cover fewer examples, penalizing the length of a rule may also be considered as another form of bias for high-coverage rules, which could also be expressed by maximizing p (or $p + n$). In any case, we also experimented with the rule length as an additional parameter. For both, linear regression and neural networks this did not lead to significant changes in the performance of the heuristics (e.g., for linear regression, the performance dropped by 0.03%).

6.2.3 Predicting the value of the final rule

Rule learning heuristics typically evaluate the quality of the current, incomplete rule, and use this measure for greedily selecting the best candidate for further refinement. However, as discussed in Sect. 6.1, if we frame the learning problem as a search problem, a good heuristic should not evaluate a candidate rule with its discriminatory power, but with its potential to be refined into a good final rule. Such a utility function could be learned with a reinforcement learning algorithm, which will learn to predict in each step of the refinement process which refinement is most likely to lead to a good final rule. Unfortunately, Burges (2006) pointed out that this approach does not work satisfactorily.

As an alternative, we applied a method which can be interpreted as an "offline" version of reinforcement learning. We simply assign each candidate rule the precision value of its final rule in one refinement process. As a consequence, in our approach all candidate rules of one refinement process have the same target value, namely the value of the rule that has eventually been selected. Because of the deletion of all final rules that do not cover any example on the test set, we decided to remove all predecessors of such rules as well. This seemed to be the best way to handle the predecessors because we would not have a reasonable value to predict. Thus, the new meta dataset contains only 77,240 examples in total.

Figure 7 shows a histogram of the observed test-set precision values for the candidate rule (immediate reward) and for the final rule that has been learned when refining this candidate (delayed reward). Clearly, in the case of delayed rewards, the frequency of simple precision values like 0, 0.5, and 1 increases, because there are much more rules that only cover a few examples.



(a) immediate reward (b) delayed reward

Fig. 7 Histogram of the frequency of observed precision values when the target signal is the test-set precision of the candidate rule (immediate reward) and when the target signal is the test-set precision of the final rule (delayed reward)

Table 8 Macro- and micro-averaged prediction errors as well as number of conditions of the theories learned by SECO for two meta-learned heuristics, both trained using delayed rewards. The second column shows the mean absolute error (MAE) of a cross-validation of the meta-learning training set

Heuristic	MAE	Average accuracy		# conditions
		Macro	Micro	
Linear regression	0.33	77.95%	80.97%	95.63
Neural network	0.35	78.37%	81.43%	53.97

Table 8 shows the accuracies of two heuristics that were learned in this setting, the first one with a linear regression and the second one with a neural network with a single node in the hidden layer. In particular the neural network outperformed the original setting (cf. Table 7) and approaches the performance of the heuristics obtained by parameter optimization (Table 3).

6.2.4 Predicting other heuristic functions

So far, we focused on directly predicting the out-of-sample precision of a rule, assuming that this would be a good heuristic for learning a rule set. However, this choice was somewhat arbitrary. Ideally, we would like to repeat this experiment with out-of-sample values for all common rule learning heuristics. In order to cut down the number of required experiments, we decided to directly predict the number of covered positive ($\hat{p}$) and negative ($\hat{n}$) examples. Then we can combine the predictions for these values with any standard heuristic h by computing $h(\hat{p}, \hat{n})$ instead of the conventional $h(p, n)$. Note that the heuristic h only gets the predicted coverages ($\hat{p}$ and $\hat{n}$) as new input, all other statistics (e.g., P,N) are still measured on the training set. This is feasible because we designed the experiments so that the training and test set are of equal size, i.e., the values predicted for $\hat{p}$ and $\hat{n}$ are predictions for the number of covered examples on an independent test set of the same size as the training set.

Table 9 compares the performance of various heuristics using the p and n values measured on the training set, and the $\hat{p}$ and $\hat{n}$ values predicted for the test set by a trained neural network. In general, the results are disappointing. For three of the five heuristics, no signifi-

Table 9 Comparison of various heuristics with training-set coverages (p, n) and coverages predicted by the neural network $(\hat{p}, \hat{n})$

Heuristic	Args	Average accuracy		# conditions
		Macro	Micro	
Accuracy	(p, n)	75.65%	78.47%	99.13
	$(\hat{p}, \hat{n})$	75.39%	78.62%	110.80
Precision	(p, n)	76.22%	79.53%	128.37
	$(\hat{p}, \hat{n})$	76.53%	80.43%	30.00
WRA	(p, n)	75.82%	79.35%	12.00
	$(\hat{p}, \hat{n})$	69.89%	75.23%	29.97
Laplace	(p, n)	76.87%	79.76%	117.00
	$(\hat{p}, \hat{n})$	76.80%	80.77%	246.80
Correlation	(p, n)	77.55%	80.91%	47.33
	$(\hat{p}, \hat{n})$	58.09%	65.35%	40.40

cant change could be observed, but for weighted relative accuracy and correlation heuristic, the performance degrades substantially.

A surprising observation is the rather low complexity of the learned theories. For instance, the heuristic *Precision* produces very simple theories when it is used with the out-of-sample predictions, and, by doing so, increases the predictive accuracy. Apparently, the use of the predicted values of $\hat{p}$ and $\hat{n}$ allows to prevent overfitting, because the predicted positive/negative coverages are never exactly 0 and therefore the overfitting problem observed with *Precision* does not occur any more. The *Laplace* heuristic shows a similar trend, but in this case the predictions result in more complex rules than the original ones.

In summary, it seems that the predictions of both the linear regression and the neural network are not good enough to yield true coverage values on the test set. A closer look at the predicted values reveals that on the one hand both regression methods predict negative coverages and that on the other hand for the region of low coverages (which is the important one) too optimistic values are predicted (for both the positive and the negative coverage). The acceptable performance is caused by a balancing of the two imprecise predictions (as observed with the two precision-like metrics) or rather by an induced bias which tries to omit the extreme values in the evaluations (which are responsible for overfitting).

6.3 Interpretation of the learned functions

In this section, we will try to interpret the learned functions by looking at the learned weights and by looking at their coverage space isometrics.

6.3.1 Coefficients of the linear regression

Table 10 shows the coefficients for three learned regression models. In the base-line experiment, three features had a significant weight: the a priori class distribution of the examples in the training data, the precision of the rule, and the true positive rate. These feature weights were significant with a p-value smaller than 2×10^{-16} computed with the *summary*-method

Table 10 Coefficients of various functions learned by linear regression

Baseline experiment							$Acc_{\text{macro}} = 77.43\%$	
P	N	$\frac{P}{P+N}$	p	n	$\frac{p}{P}$	$\frac{n}{N}$	$\frac{p}{p+n}$	const.
0.0001	0.0001	0.7485	−0.0001	−0.0009	0.165	0.0	0.3863	0.0267
Delayed reward scenario							$Acc_{\text{macro}} = 77.59\%$	
P	N	$\frac{P}{P+N}$	p	n	$\frac{p}{P}$	$\frac{n}{N}$	$\frac{p}{p+n}$	const.
0.0	0.0002	0.8772	−0.0002	0.0002	0.2103	−0.297	0.1367	0.2282
Delayed reward + logarithmic coverage							$Acc_{\text{macro}} = 78.88\%$	
$\log(P+1)$	$\log(N+1)$	$\frac{P}{P+N}$	$\log(p+1)$	$\log(n+1)$	$\frac{p}{P}$	$\frac{n}{N}$	$\frac{p}{p+n}$	const.
0.0709	−0.0255	0.0521	0.1139	−0.0588	0.1379	−0.3673	−0.1032	0.427

of R.[4] Only the false positive rate was not statistically significant. At first it may be surprising that the false positive rate is not significant, but its main role is to ensure consistency, which can—in the regions of interest—also be ensured with precision. Thus, if we only consider feature weights above 0.1 as important for the model, the learned heuristic linearly combines class distribution, coverage and consistency. Informally, we can also observe that, in line with our observations from Sect. 5, consistency receives a higher weight than coverage, although it is not entirely clear whether these values are directly comparable.

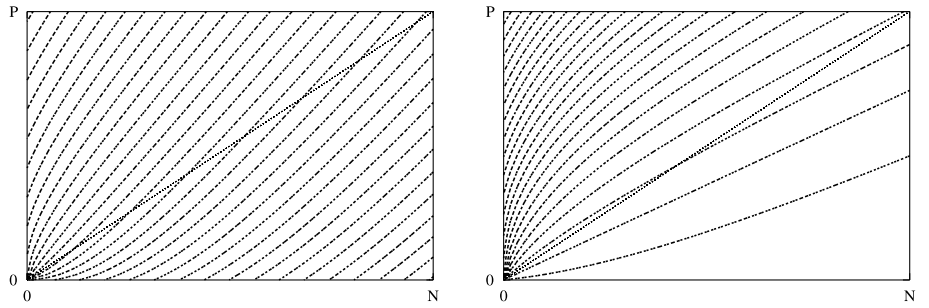
This can be more clearly seen from the coefficients learned in the delayed reward scenario, where the function was trained on the test set precision of the best refinement of the rule. The function is quite similar to the previous one, except that the consistency is now enforced through two factors: a high negative weight on the false positive rate and a positive weight on precision. In this scenario the weight of the total positives was only significant at $p = 0.01$, all others were also significant with a p-value of at least 3.69×10^{-5}.

In both cases, the current coverage of a rule (p and n) and the total example counts of the data (P and N) have comparably low weights. This is not that surprising if one keeps in mind that the target value is in the range [0, 1], while the absolute values for p and n are in a much higher range. We nevertheless included them because we believe that in particular for rules with low coverage, the absolute numbers are more important than their relative fractions. A rule that covers only a single example will typically be bad, irrespective of the size of the original dataset.

In the light of these results, we made two more experiments: In the first, we removed the four coverage values from the input, and learned another function from the remaining four features. This did not change the performance very much (77.20% macro-averaged accuracy).

In a second experiment, we used the logarithmic values $\log(P+1)$, $\log(N+1)$, $\log(p+1)$, $\log(n+1)$ instead, with the idea that the importance of differences in coverage is proportional to the coverage. This considerably improved the results for linear regression. The last part of Table 10 shows the learned function. There are a few interesting differences to the previous functions: (i) the logarithmic coverage values get a much higher weight than their absolute counterparts (all significant at $p < 2 \times 10^{-16}$), (ii) the prior class probability

[4] http://www.r-project.org/.



(a) Linear Regression (b) Neural Network

Fig. 8 Isometrics of heuristics meta-learned with linear regression and a neural network in the delayed reward scenario

$P/(P+N)$ receives a much lower weight (still significant with $p = 0.0021$), and (iii) precision receives now a negative weight (also significant at $p < 2 \times 10^{-16}$), which is presumably counterbalanced by the much higher negative weight on the false positive rate.

6.3.2 Isometrics of the heuristics

To understand the behavior of the learned heuristics, we will again take a look at their isometrics in coverage space. Figure 8 shows isometrics of the heuristic learned in the experiment with delayed rewards (without the logarithmic features) in a coverage space with 60×48 examples (the sizes were chosen arbitrarily but are the same as for all other previous isometrics). The left part of the figure displays the isometrics of the heuristic that was learned by linear regression on the dataset that used only the relative features (see Sect. 6.3.1). The right part shows the best-performing neural network (the one that uses only one node in the hidden layer).

Apparently, both functions learn somewhat different heuristics. Superficially, the isometrics of the linear regression heuristic are quite similar to the parallel lines of the cost heuristic, but, just as we observed in the experiments of Sect. 5 (cf. Fig. 6 (d)), their slope is generally > 1, i.e., false positives are weighted more heavily than true positives. The isometrics for the neural net seems to employ a trade-off similar to those of the F-measure. The shift towards the N-axis is reminiscent of the F-measure (cf. Fig. 2), which tries to correct the undesirable property of precision that all rules that cover no negative examples are evaluated equally, irrespective of the number of positive examples that they cover. Interestingly, the isometrics of the linear regression function with logarithmic features (not shown) have a quite similar appearance.

However, in all cases the isometrics have a non-linear shape, which bends them towards the N-axis when they approach the P-axis. Thus, in regions with high consistency, the bias that prefers consistency over coverage is even more emphasized. This also has a somewhat surprising effect, namely a small bias towards rules that cover a low number of positive examples (compared to regular precision). Intuitively, one would expect the opposite, namely that rules with low coverage are avoided because they are likely to be unreliable and noisy. This confirms our results for the Klösgen measure, where we could see that parameter values $\omega > 1$ encode a bias that avoids low coverage regions (cf., e.g., the graph for $\omega = 2$ in Fig. 3), but that these values did not perform well empirically. In some sense, this may be interpreted as support for the well-known *small disjuncts problem*, first observed by Holte

et al. (1989), namely that rules with low coverage contribute significantly to the overall error of a rule set, but that they also cannot be omitted without a loss in accuracy.

7 Discussion

The crucial step for the performance of a greedy covering algorithm is the choice of the rule evaluation heuristic. In this paper, we have tried to empirically determine a good trade-off between consistency and completeness, which are the most important criteria for evaluating the quality of a rule.

However, it is important to keep in mind that a good rule must optimize or trade off various criteria simultaneously. Among them are:

Consistency: How many negative examples are covered?
In concept learning or decision list learning, if a rule covers negative examples, these misclassifications cannot be corrected with subsequent rules. When sets of unordered rules are learned for multi-class problems, such corrections are possible, but obviously one should nevertheless try to find pure rules.

Completeness: How many positive examples are covered?
Even though subsequent rules may cover examples that the current rule leaves uncovered, rule with higher coverage are typically preferred because they bring the learner closer to the goal of covering all positive examples.

Gain: How good is the rule in comparison to other rules (e.g., default rule, predecessor rules)?
A rule with high consistency may be bad if its predecessor had a higher consistency and vice versa. Thus, many heuristics, such as weighted relative accuracy, or information gain relate the quality of a rule to other rules.

Utility: How useful will the rule be in the context of the other rules in the theory?
A rule with high consistency and completeness may nevertheless be a bad addition to the current theory if it does not explain any new examples.

Bias: How will the quality estimate change on new examples?
It is well-known that estimates obtained on the training data will be optimistically biased. A good heuristic has to address this problem so that the algorithm will not overfit the training examples.

Potential: How close is the current rule to a good rule?
An incomplete rule, i.e., a rule that is encountered during the search for a good rule, should not be evaluated by its ability to discriminate between positive and negative examples, but by its potential to be refined into such a rule.

Simplicity: How complex or comprehensible is the rule?
In addition to its predictive quality, rules are often also assessed by their comprehensibility, because this is one of the key factors for preferring rule learning algorithms over competing inductive classification algorithms. As comprehensibility is difficult to measure, it is often equated with simplicity or rule length.

Our experiments addressed many of this issues directly (consistency, completeness, gain, bias). Some of these criteria are also addressed, at least to some extent, algorithmically. Utility, for example, is addressed in part by the covering loop which removes examples that

have been covered by previous rules. Thus, the context of past rules is taken into account. However, it is harder to take the context of rules that will be subsequently learned into account. This is particularly the case because most rule learning heuristics only focus on the examples covered by the rule, and not on the examples that are not covered by a rule. This is contrary to decision tree learning heuristics, which consider all possible outcomes of a condition simultaneously.[5] The Part (Frank and Witten 1998) algorithm may be viewed as an attempt to use the best of both worlds. Similarly, Ripper's global optimization phase, where rules in a final theory are tentatively re-learned in the context of all previous and all subsequent rules, may be viewed as an attempt to address this issue.

It is also important to realize that these criteria are not independent. For example, comprehensibility and simplicity are correlated with completeness: simple rules tend to be more general and to cover more examples. Thus, a bias for completeness will automatically correlate with a bias for shorter rules. Similarly, the idea of the Laplace-correction as introduced in CN2 (Clark and Boswell 1991) was to on the one hand correct a too strong bias for consistency over completeness (by, e.g., penalizing pure rules that cover only single examples), and on the other hand also to try to provide more accurate probability estimates.

Nevertheless, it is not clear whether all these points can or should be addressed simultaneously with a single rule learning heuristic. Answering this question is beyond the scope of this paper (in fact, we do believe that these problems should be separated and addressed individually). However, it is the case that common rule learning algorithms essentially assume that all these objectives can be captured into a single heuristic function (only overfitting is frequently addressed by using a separate criterion). Thus, it is a valid and interesting question how good a greedy rule learner can get under this assumption. The work reported in this paper may be viewed as a contribution to answering this question. It is, in our opinion, the necessary first step to a systematic investigation of heuristic rule learning.

8 Related work

While there are several empirical comparisons of splitting heuristics for decision tree induction (Mingers 1989; Buntine and Niblett 1992), there are, somewhat surprisingly, relatively few works that empirically compare different rule learning heuristics. For example, Lavrač et al. (1992a, 1992b) compare several heuristics for inductive logic programming. Most works only perform a fairly limited comparison, which typically introduces a new heuristic and compares it to the heuristic used in an existing system. A typical example for work in this area is (Todorovski et al. 2000), where the performance of weighted relative accuracy was compared to the performance of CN2's Laplace-heuristic. To our knowledge, our work reported in this paper is the most exhaustive empirical work in this respect.

On the other hand, considerable progress has been made in the principal understanding of rule learning heuristics. As discussed in Sect. 3.1, Fürnkranz and Flach (2005) have introduced coverage space isometrics as a means for visualizing rule evaluation metrics. Using this tool, they have derived several interesting results, such as that the m-estimate effectively trades off precision and weighted relative accuracy. While their paper contributed to a better understanding of rule learning heuristics, the authors concluded that, in general, rule learning heuristics are not yet well understood.

[5] Consider, e.g., the difference between the information gain heuristic used in ID3 (Quinlan 1983) and the information gain heuristic used in Foil (Quinlan 1990).

There has also been significant progress on analyzing rule evaluation metrics that are commonly used in descriptive induction tasks such as association rule discovery or subgroup discovery. Most notably, Tan et al. (2002) have surveyed 21 rule learning heuristics and compared them according to a set of desirable properties. In general, they conclude that the choice of the right interestingness measure is application-dependent, but they also identify situations in which many measures are highly correlated with each other. Bayardo Jr. and Agrawal (1999) analyze several heuristics in support and confidence space, and show that the optimal rules according to many criteria lie on the so-called support/confidence border, the set of rules that have maximum or minimum confidence for a given support level. Recently, Wu et al. (2007) showed that a group of so-called null-invariant measures (measures that are not influenced by the number of records that do not match the pattern) can be generalized into a single parametrized heuristic. We plan to analyze this parametrized heuristic with the apparatus that we have used for our results in Sect. 5.

Naturally, there are some similarities between heuristics used for descriptive and for predictive tasks. For example, Lavrač et al. (1999) derived weighted relative accuracy in an attempt to unify these two realms, or Fürnkranz and Flach (2004) analyzed filtering and stopping heuristics and showed that Foil's information gain search and MDL-based pruning has a quite similar effect as support and confidence thresholds that are commonly used in association rule discovery. Nevertheless, it is important to note that good heuristics for descriptive induction are not necessarily well-suited for predictive induction (weighted relative accuracy is a good example). The key difference is that in the latter case one typically needs to learn an entire rule set, where lack of coverage in individual rules can be corrected by the entire ensemble of rules. Inconsistencies, on the other hand, cannot be corrected by the induction of additional rules (at least not in the case of concept learning). In this light, the result of this paper, that good heuristics for predictive induction will favor consistency over coverage, appears to be reasonable.

Our results may also be viewed in the context of trying to correct overly optimistic training error estimates (resubstitution estimates). In particular, in some of our experiments, we try to directly predict the out-of-sample precision of a rule. This problem has been studied theoretically by Scheffer (2005) and Mozina et al. (2006). In other works, it has been addressed empirically. For example Vapnik et al. (1994) have used empirical data to measure the VC-Dimension of learning machines. Fürnkranz (2004) also creates meta data in a quite similar way, and tries to fit various functions to the data. But the focus there is the analysis of the obtained predictions for out-of-sample precision, which is not the key issue in our experiments.

9 Conclusions

The experimental study reported in this paper has provided several important insights into the behavior of greedy inductive rule learning algorithms.

First, we think that this has been the most exhaustive experimental comparison of different rule learning heuristics to date. We tested five parameter-free heuristics, five parametrized heuristics with a large number of parameterizations, and several different metalearning scenarios. The results confirm several previously known findings (e.g., precision and Laplace overfit, whereas accuracy and weighted relative accuracy over-generalize), but also yielded new insights into their comparative performance. In particular, we have determined suitable default values for commonly used parametrized evaluation metrics such as the m-estimate. This is of considerable practical importance, as we showed that these new

values outperformed conventional search heuristics and performed comparably to the Ripper rule learning algorithm. On the other hand, however, we have also seen some indication that these values capture aspects of our algorithm that may not fully transfer to other rule learning algorithms.

Second, our results also let us draw important conclusions about what factors influence a good performance of a rule learning heuristic. For example, we found that heuristics which take the a priori class distribution into account (e.g., by evaluate relative coverage instead of absolute coverage) will in general outperform heuristics that ignore the class distribution (e.g., the F-measure which trades off recall and precision). This is also confirmed by the high weight that this parameter receives in our meta-learned heuristics. Gain heuristics, which take this one step further by considering the distribution of the predecessor rule as a prior class distribution for each individual addition to a rule, may be even more preferable, but this still needs to be investigated more thoroughly.

We also found that for a good overall performance, it is necessary to prefer consistency over coverage, i.e., to weight the false positive rate more heavily than the true positive rate. We can most clearly observe this bias towards minimizing the false positive rate in the optimal parameter value for the relative cost metric, but it can also be observed in other well-performing heuristics whose isometrics have a very steep slope in the important regions. In the experiments with metalearning and in the good performance of the correlation heuristic we can also observe that heuristics perform better if they increase the emphasis on this aspect for rules with high consistency.

This result may also be interpreted as evidence that a good heuristic has to adapt to the characteristics of the algorithm in which it is used. In our case, this bias towards consistency seems to be a desirable property for a heuristic that is used in a covering algorithm, where incompleteness (not covering all positive examples) is less severe than inconsistency (covering some negative examples), because incompleteness can be corrected by subsequent rules, whereas inconsistency cannot (at least not in a concept learning scenario). This dependency on the dynamics of the algorithm is also confirmed by one of the results of the metalearning study, in which we observed that training on the test-set performance of the candidate rule is somewhat less efficient than training on the performance of its best refinement. Finally, the results of the transfer of the heuristics to other rule learning implementations also seem to confirm that they capture individual characteristics of the algorithm.

However, our results also have their limitations. For example, we have only evaluated the overall performance over a wide variety of datasets. Obviously, we can expect a better performance if the parameter values are tuned to each individual dataset. We think that the good performance of Ripper is due to the flexibility of post-pruning, which allows to adjust the level of generality of a rule to the characteristic of a particular dataset. We have deliberately ignored the possibility of pruning for this set of experiments, because our goal was to gain a principal understanding of what constitutes a good rule evaluation metric for separate-and-conquer learning. We are currently investigating the interplay of pruning and learning in more detail.

Acknowledgements We would like to thank the reviewers for very helpful comments that improved the paper considerably. We would also like to thank Peter Clark for making CN2 publicly available, and the group in Waikato for their work on *Weka*.

This research was supported by the *German Science Foundation (DFG)* under grant No. FU 580/2.

References

Akaike, H. (1974). A new look at the statistical model selection. *IEEE Transactions on Automatic Control*, *19*(6), 716–723.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://archive.ics.uci.edu/ml/.

Bayardo, R. Jr., & Agrawal, R. (1999). Mining the most interesting rules. In *Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining (KDD-97)* (pp. 145–154).

Brin, S., Motwani, R., & Silverstein, C. (1997). Beyond market baskets: generalizing association rules to correlations. In *Proceedings of the ACM SIGMOD international conference on management of data* (pp. 265–276).

Buntine, W., & Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, *8*, 75–85.

Burges, S. (2006). Meta-Lernen einer Evaluierungs-Funktion für einen Regel-Lerner. Master's thesis, TU Darmstadt, December 2006 (in German) (English title: *Meta-learning of an evaluation function for a rule learner*).

Cestnik, B. (1990). Estimating probabilities: a crucial task in machine learning. In L. Aiello (Ed.), *Proceedings of the 9th European conference on artificial intelligence* (pp. 147–150). ECAI-90, Stockholm, Sweden, 1990. London: Pitman.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European working session on learning* (pp. 151–163). EWSL-91, Porto, Portugal, 1991. Berlin: Springer.

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

Cohen, W. W. (1995). Fast effective rule induction. In A. Prieditis & S. Russell (Eds.), *Proceedings of the 12th international conference on machine learning* (pp. 115–123). Tahoe City, CA, July 9–12, 1995. San Mateo: Morgan Kaufmann.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, *7*, 1–30.

Fan, R.-E., Chen, P.-H., Lin, C.-J., & Joachims, T. (2005). Working set selection using the second order information for training SVM. *Journal of Machine Learning Research*, *6*, 1889–1918.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In J. Shavlik (Ed.), *Proceedings of the 15th international conference on machine learning* (pp. 144–151). ICML-98, Madison, WI, 1998. San Mateo: Morgan Kaufmann.

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

Fürnkranz, J. (2004). Modeling rule precision. In J. Fürnkranz (Ed.), *Proceedings of the ECML/PKDD-04 workshop on advances in inductive rule learning* (pp. 30–45). Pisa, Italy, 2004.

Fürnkranz, J. (2004). FOSSIL: A robust relational learner. In F. Bergadano & L. De Raedt (Eds.), *Lecture notes in artificial intelligence: Vol. 784. Proceedings of the 7th European conference on machine learning* (pp. 122–137). ECML-94, Catania, Italy, 1994. Berlin: Springer.

Fürnkranz, J. (1997). Pruning algorithms for rule learning. *Machine Learning*, *27*(2), 139–171.

Fürnkranz, J., & Flach, P. (2004). An analysis of stopping and filtering criteria for rule learning. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.), *Lecture notes in artificial intelligence: Vol. 3201. Proceedings of the 15th European conference on machine learning* (pp. 123–133). ECML-04, Pisa, Italy, 2004. Berlin: Springer.

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

Fürnkranz, J., & Widmer, G. (1994). Incremental reduced error pruning. In W. Cohen & H. Hirsh (Eds.), *Proceedings of the 11th international conference on machine learning* (pp. 70–77). ML-94, New Brunswick, NJ, 1994. San Mateo: Morgan Kaufmann.

Holte, R., Acker, L., & Porter, B. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the 11th international joint conference on artificial intelligence* (pp. 813–818). IJCAI-89, Detroit, MI, 1989. San Mateo: Morgan Kaufmann.

Janssen, F., & Fürnkranz, J. (2007). On meta-learning rule learning heuristics. In *Proceedings of the 7th IEEE conference on data mining* (pp. 529–534). ICDM-07, Omaha, NE, 2007.

Janssen, F., & Fürnkranz, J. (2008). An empirical investigation of the trade-off between consistency and coverage in rule learning heuristics. In T. Horvath, J.-F. Boulicaut, & M. Berthold (Eds.), *Proceedings of the 11th international conference on discovery science* (pp. 40–51). DS-08, Budapest, Hungary, 2008. Berlin: Springer.

Janssen, F., & Fürnkranz, J. (2009). A re-evaluation of the over-searching phenomenon in inductive rule learning. In *Proceedings of the SIAM international conference on data mining* (pp. 329–340). SDM-09, Sparks, NV, 2009.

Klösgen, W. (1992). Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. *International Journal of Intelligent Systems*, *7*, 649–673.

Lavrač, N., Flach, P., & Zupan, B. (1999). Rule evaluation measures: a unifying view. In S. Džeroski & P. Flach (Eds.), *Proceedings of the 9th international workshop on inductive logic programming (ILP-99)* (pp. 174–185). Berlin: Springer.

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

Lavrač, N., Cestnik, B., & Džeroski, S. (1992a). Search heuristics in empirical inductive logic programming. In *Logical approaches to machine learning, workshop notes of the 10th European conference on AI*, Vienna, Austria, 1992.

Lavrač, N., Cestnik, B., & Džeroski, S. (1992b). Use of heuristics in empirical inductive logic programming. In S. H. Muggleton & K. Furukawa (Eds.), *Proceedings of the 2nd international workshop on inductive logic programming (ILP-92)*, Number TM-1182 in ICOT Technical Memorandum, Tokyo, Japan, 1992. Institute for New Generation Computer Technology.

Michalski, R. S. (1969). On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th international symposium on information processing* (pp. 125–128). Switching Circuits, Vol. A3, FCIP-69, Bled, Yugoslavia, 1969.

Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, *3*, 319–342.

Mozina, M., Demšar, J., Zabkar, J., & Bratko, I. (2006). Why is rule learning optimistic and how to correct it. In *Machine learning: ECML 2006, 17th European conference on machine learning* (pp. 330–340).

Muggleton, S. H. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*(3, 4), 245–286. Special issue on inductive logic programming.

Quinlan, J. (1996). Learning first-order definitions of functions. *Journal of Artificial Intelligence Research*, *5*, 139–161.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*, 239–266.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning. An artificial intelligence approach* (pp. 463–482). Palo Alto: Tioga.

Salton, G., & McGill, M. J. (1986). *Introduction to modern information retrieval*. New York: McGraw-Hill.

Scheffer, T. (2005). Finding association rules that trade support optimally against confidence. *Intelligent Data Analysis*, *9*(3), 381–395.

Tan, P.-N., Kumar, V., & Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. In *Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 32–41). KDD-02, Edmonton, Alberta, 2002.

Thiel, M. (2005). Separate and Conquer Framework und disjunktive Regeln. Master's thesis, TU Darmstadt, 2005. In German (English title: *Separate and conquer framework and disjunctive rules*).

Todorovski, L., Flach, P., & Lavrac, N. (2000). Predictive performance of weighted relative accuracy. In D. A. Zighed, J. Komorowski, & J. Zytkow (Eds.), *4th European conference on principles of data mining and knowledge discovery (PKDD2000)* (pp. 255–264). Berlin: Springer.

Vapnik, V., Levin, E., & Cun, Y. L. (1994). Measuring the VC-dimension of a learning machine. *Neural Computation*, *6*(5), 851–876.

Witten, I. H., & Frank, E. (2005). *Data mining—practical machine learning tools and techniques with java implementations* (2nd edn.). San Mateo: Morgan Kaufmann. http://www.cs.waikato.ac.nz/~ml/weka/.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In J. Komorowski & J. Zytkow (Eds.), *Proc. first European symposium on principles of data mining and knowledge discovery* (pp. 78–87). PKDD-97, Berlin, 1997. Berlin: Springer.

Wu, T., Chen, Y., & Han, J. (2007). Association mining in large databases: a re-examination of its measures. In *Proceedings of the 11th European symposium on principles of data mining and knowledge discovery* (pp. 621–628). PKDD-07, Warsaw, Poland, 2007. Berlin: Springer.

Xiong, H., Shekhar, S., Tan, P.-N., & Kumar, V. (2004). Exploiting a support-based upper bound of Pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 334–343). KDD-04, Seattle, USA, 2004.