Technische Universität Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10, D-64289 Darmstadt, Germany

http://www.ke.informatik.tu-darmstadt.de

Technical Report TUD–KE–2007–02

Frederik Janssen, Johannes Fürnkranz

Meta-Learning Rule Learning Heuristics

Meta-Learning Rule Learning Heuristics

Frederik Janssen janssen@ke.informatik.tu-darmstadt.de
Johannes Fürnkranz fuernkranz@informatik.tu-darmstadt.de
Knowledge Engineering Group
Department of Computer Science
TU Darmstadt, Germany

Abstract

The goal of this paper is to investigate to what extent a rule learning heuristic can be learned from experience. Our basic approach is to learn a large number of rules and record their performance on the test set. Subsequently, we train regression algorithms on predicting the test set performance from training set characteristics. We investigate several variations of this basic scenario, including the question whether it is better to predict the performance of the candidate rule itself of the resulting final rule. Our experiments on a number of independent evaluation sets show that the learned heuristics outperform standard rule learning heuristics. We also analyze their behavior in coverage space.

1. Introduction

The long-term goal of our research is to understand the properties of a heuristic that will perform well in a broad selection of rule learning algorithms. 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, Flach, & Zupan, 1999), which proposed weighted relative accuracy as a novel heuristic, and (Fürnkranz & Flach, 2005), in which a wide variety of rule evaluation metrics were anlyzed and compared by visualizing their behavior in ROC space. There are also some works on comparing properties of association rule evaluation measures (e.g., (Tan, Kumar, & Srivastava, 2002)) but these have different requirements than classification rules (e.g., completeness is not an issue there).

Recently, (Janssen & Fürnkranz, 2006) performed an experimental comparison of commonly used rule learning heuristics, and, in particular, found parameter settings for three parametrized heuristics, which performed quite well on a large number of datasets. Interestingly, the authors observed that these values resulted in heuristics with very similar behavior. Nevertheless, the shape of these heuristics was predetermined.

In this work, we take a different road to identifying a good search heuristic for classification rule learning algorithms. The key idea is to meta-learn such a heuristic from experience, without a bias towards existing measures. Consequently, we created a large meta data set (containing information from which we assume that the "true" performance of a rule can be learned) and perform a regression with various methods on it. On this dataset, we learned an evaluation function and used it as a search heuristic inside our implementation of a simple rule learner. We experimented with various options for generating the meta datasets, tried to assess the importance of features, experimented with different

meta-learning algorithms, and also tried a setting in which the learner tries to predict the performance of a *complete* rule from its incomplete predecessors.

The ruler learner, which is used for generating the meta data and for evaluating the learned heuristics, is described in Section 2, after which we continue with a brief discussion of rule learning heuristics (Section 3). The meta data generation and the experimental setup is described in Section 4. The main results are presented in Section 5.

2. Rule Learning Algorithm

For the purpose of this empirical study, we implemented a simple *Separate-and-conquer* or *Covering* rule learning algorithm (Fürnkranz, 1999) within the *Weka* machine learning environment (Witten & Frank, 2005). Both the outer loop (the covering procedure) and the top-down refinement inside the learner are fairly standard. For details about the implementation see (Fürnkranz, 2004, 1999).

Separate-and-conquer rule learning can be divided into two main steps: First, a rule is learned from the training data by a greedy search (the *conquer* step). Second, all examples covered by the learned rule are removed from the data set (the *separate* step). Then, the next rule is learned on the remaining examples. Both steps are repeated as long as positive examples are left in the training set. The refinement procedure, which is used inside the conquer step of the algorithm, returns all possible candidate refinements that can be obtained by adding a single condition to the body of the rule. All refinements are evaluated with a heuristic, and the best rule is selected.

Our implementation continues to greedily refine the current rule until no negative example is covered any more. In this case, the search stops and the best rule encountered during the refinement process is added to the theory. Thus, the best rule is not necessarily the last one searched. A small optimization was to stop the refinement process when no refinement could possibly achieve a better heuristic evaluation than the current best rule, i.e., when a hypothetical refinement that covers all remaining positive examples and no negative example achieves a lower evaluation than the best rule found so far. We used random tie breaking for rules with equal evaluation, and filtered out candidate rules that do not cover any positive examples. Rules were added to the theory until a new rule would not increase the accuracy of the theory on the training set (this is the case when the learned rule covers more negative than positive examples).

We did not use any specific pruning technique, but solely relied on the evaluation of the rules by the used rule learning heuristic. Note, however, that this does not 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.

3. Rule Learning Heuristics

Numerous heuristics have been provided for inductive rule learning, a general survey can be found in (Fürnkranz, 1999). Most rule learning heuristics can be seen as functions of the following four arguments:

- P and N: the number of positive/negative examples in the training set

Table 1: Search heuristics used in this study

heuristic	function
precision	$\frac{p}{p+n} \sim \frac{p-n}{p+n}$
Laplace	$\frac{p+1}{p+n+2}$
accuracy	$\frac{p+(N-n)}{P+N} \sim p-n$
WRA	$\frac{p+n}{P+N}(\frac{p}{p+n} - \frac{P}{P+N}) \sim \frac{p}{P} - \frac{n}{N}$
correlation	$\frac{p(N-n)-(P-p)n}{\sqrt{PN(p+n)(P-p+N-n)}}$

- p and n: the number of positive examples covered by the rule

Examples of heuristics of this type are the commonly used heuristics that are shown in Figure 1. Precision is known to overfit the data, weighted relative accuracy (Todorovski, Flach, & Lavrac, 2000) has a tendency to over-generalize. (Fürnkranz & Flach, 2005) have shown that the Laplace heuristic and its generalization, the m-estimate which is defined by $\frac{p+m\cdot\frac{P}{P+N}}{p+n+m}$, form a trade-off between these two extremes. In (Janssen & Fürnkranz, 2006), a parameter value for the m-estimate was determined that optimizes this trade-off. The correlation heuristic also has a very good overall performance (Fürnkranz, 1994).

As P and N are constant for a given learning problem, these heuristics effectively only differ in the way they trade off completeness (maximizing p) and consistency (minimizing n), and may thus be viewed as a function $h(p, n)$. As a consequence, each rule may be viewed as a point in coverage space, a variant of ROC space that uses the absolute numbers of true positives and false positives as its axes. The preference bias of different heuristics may then be visualized by plotting the respective heuristic values of the rules on top their locations in coverage space, resulting in a 3-dimensional plot $(p, n, h(p, n))$. 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. (Fürnkranz & Flach, 2005) have proposed this technique for analyzing the behavior of rule learning heuristics. Another method is to plot both contour lines and the surface of the function which is done in our visualization (cf. Section 5.3).

The goal of our work is to automatically learn such a function $h(p, n)$, which allows to predict the quality of a learned rule. However, note that most of the functions in Table 1 contain some 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 meta-learning 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

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

Some heuristics use additional components, such as

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

We will evaluate the utility of taking the rule's length into account. However, as our goal is to find a function that allows to evaluate a rule irrespective of how it has been learned, we will not consider the parameters p' and n'. Note that 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.

4. Meta-Learning Scenario

4.1 Definition of the Meta-Learning Task

The key issue for our work is how to define the meta-learning 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 completed the learning of a rule. After learning a complete theory, the learner receives a reinforcement signal (e.g., the estimated accuracy of the learned theory), 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 approach: 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, or we can use the performance of its best refinement (we evaluated both approaches). The latter is described in Section 5.6. In order to assess the performance of a rule, we used its out-of-sample precision, but, again, we have also experimented with other choices.

4.2 Meta Data Generation

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. Therefore, we used the rule learner described above for obtaining the above-mentioned characteristics for each learned rule. These form a training instance in the meta data set. The training signals are the performance parameters of the rule on the test set.

```
procedure 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 pos than negs
  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(Rule,Positive(TrainSet))|
    n ← |Covered(Rule,Negative(TrainSet))|
    P ← |Covered(Rule,TotalNegative(TrainSet))|
    N ← |Covered(Rule,TotalNegative(TrainSet))|
    l ←Length(Rule)
    p̂ ← |Covered(Rule,Positive(TestSet))|
    n̂ ← |Covered(Rule,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)
```

Figure 1: Algorithm for generating the Meta Data

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. The GenerateMetaData procedure described in Figure 1 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: We used 27 datasets with varying characteristics (different number of classes, attributes, instances) from the UCI Repository (Newman, Blake, Hettich, & Merz, 1998).[1]

5x2 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.

Classes: For each dataset and each fold, we generated one dataset for each class, treating this class as the positive one 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 binary datasets, each time using a different search heuristic. We used all of the heuristics of Table 1.The first four form a representative selection of search heuristics with linear ROC space isometrics (Fürnkranz & Flach, 2003), while the correlation heuristic has non-linear isometrics. These heuristics represent a large variety of learning biases. For example, it is known that *WRA* and *Accuracy* tend to prefer simpler rules with high coverage, whereas *Precision* and *Laplace* show a tendency to learn possibly complex rules with high precision on the training set.

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

4.3 Regression Methods

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 non-linear terms that are used as basic features (e.g., $p/(p+n)$).

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 used *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 did not improve.

1. 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

Table 2: Accuracies for several methods

method	MAE	Accuracy	# conditions
LinearRegression	0.22	77.43%	117.6
MLP (1 node)	0.28	77.81%	121.3
MLP (5 nodes)	0.27	77.37%	1085.8
MLP (10 nodes)	0.27	77.53%	112.7

Both algorithms are provided by *Weka* (Witten & Frank, 2005) and were initialized with standard parameters.

4.4 Evaluation methods

Our primary method for evaluating the introduced heuristics is to use these heuristics inside the rule learner. We evaluated the heuristics on 30 UCI data sets[2] which were not used during the training phase. Like the 27 data sets on which the rules for the meta data are induced, these 30 sets have varying characteristics to ensure that our method will perform well under a wide variety of conditions. On each dataset, the rule learner with the learned heuristics was evaluated with one iteration of a 10-fold cross validation. The performance over all sets was then averaged. We also evaluated the length of the theories in terms of number of conditions.

The fit of the learned functions to the target values can also be evaluated in terms of the mean absolute error, again estimated by one iteration of a 10-fold cross validation on the training data.

$$MAE(f') = \frac{1}{m}\sum_{i=0}^{m}|f'(i) - f(i)|$$

with m denotes the number of instances, $f(i)$ the actual value, and $f'(i)$ the predicted value of instance i. The mean absolute error measures the error made by the regression model on unseen data. Therefore it provides no clear insight into the functionalities when it is used as heuristic by the rule learner. Hence, a low mean absolute error on the meta data set does not implicate that the function works good as heuristic (cf. Table 2).

5. Results

5.1 Quantitative Results

In the first experiment, we wanted to see how accurately we can predict the out-of-sample precision of a rule. We trained a linear regression model and a neural network on the eight measurements that we use for characterizing a rule (cf. Section 3) using the precision values measured on the test sets as a target function. Table 2 displays results for the Linear Regression and 3 different neural networks, with different numbers of nodes in the hidden layer. The performances of the four algorithms are quite comparable, with the possible

2. 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

Table 3: Coefficients of the Linear Regression

P	N	$\frac{P}{P+N}$	p	n	$\frac{p}{P}$	$\frac{n}{N}$	$\frac{p}{p+n}$	constant
0.0001	0.0001	0.7485	-0.0001	-0.0009	0.165	0.0	0.3863	0.0267

exception of the neural network with 5 nodes in the hidden layer. This induced very large theories (over 1000 conditions on average), and also had a somewhat worse performance in predictive accuracy. As discussed in Section 4.4, a low mean absolute error does not necessarily imply an accurate heuristic as becomes obvious when considering Table 2.

5.2 Coefficients of the Linear Regression

It is interesting to have a look at the learned concepts. Table 3 shows the coefficients of the learned regression model. The most important feature was the *a priori* distribution of the examples in the training data followed by the precision of the rule. Interestingly, while the *tpr* has a non-negligible influence on the result, the *fpr* is practically ignored.

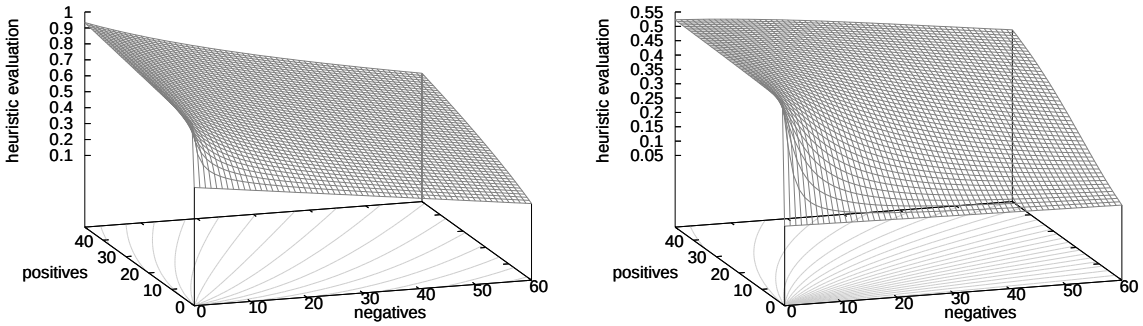
Both the current coverage of a rule (p and n) and the total example counts of the data (P and N) have comparably low weights This is not that surprising if one keeps in mind that the target value is in the range $[0, 1]$, while the absolute values for p and n are in a much higher range. We nevertheless had 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 order to see whether we can completely ignore the absolute values, we learned another function which only used $\frac{P}{P+N}$, p/P, n/N and $\frac{p}{p+n}$ as input values. The linear regression function trained on this dataset performed insignificantly worse than the one that is computed on the original set (77.43% accuracy vs. 77.20% accuracy). For the neural networks, the performance degradation was somewhat worse.

5.3 Isometrics of the Heuristics

To understand the behavior of the learned heuristics, we follow the framework of (Fürnkranz & Flach, 2005) and analyze their isometrics in ROC or coverage space. Figure 2 shows a 3d-plot of the surface of the learned heuristic in a coverage space with 60x48 examples (the sizes were chosen arbitrarily). The bottom of the graph, shows isometric lines that characterize this surface. The upper part of the figure displays the isometrics of the heuristic that was learned by linear regression on the data set that used only the relative features (see Section 3). The lower 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. Although the 3d-surfaces looks fairly similar to each other, the isometric lines reveal that the learned heuristics are, in fact, quite different. Those for the linear regression are like a variant of weighted relative accuracy, but with a different cost model (i.e. false negatives are more costly than false 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 (for an illus-



(a) Linear Regression (b) Neural Network

Figure 2: Isometrics of the two functions (immediate precision)

tration see (Janssen & Fürnkranz, 2006)), 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.

However, both heuristics have a non-linear shape of the isometrics in common, which bends the lines towards the N-axis. Effectively, this encodes a bias towards rules that cover a low number of positive examples (compared to regular precision). This 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.

The isometrics of the linear regression are somewhat curious. In areas of high positive coverage they behave like those of the neural network but not with that strong bend towards the N-axis. The isometrics are symmetric which leads to a similar effect observed at the neural net. Thus, in areas with high negative coverage rules are preferred that cover a low number of negative examples.

5.4 Including the Length of the Rules

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. As we will see later on the data set with the 4 features derived in Section 5.2 are sufficient to learn a good heuristic with the Linear Regression.

Table 4: Comparison of various heuristics with training-set (p, n) and predicted $(\hat{p}, \hat{n})$ coverages

heuristic	args	Accuracy	# conditions
Accuracy	(p, n)	75.60%	104.77
	$(\hat{p}, \hat{n})$	75.39%	110.8
Precision	(p, n)	76.22%	129.17
	$(\hat{p}, \hat{n})$	76.53%	30.0
WRA	(p, n)	75.80%	12.13
	$(\hat{p}, \hat{n})$	69.89%	29.97
Laplace	(p, n)	76.89%	118.83
	$(\hat{p}, \hat{n})$	76.80%	246.8
Correlation	(p, n)	77.57%	47.5
	$(\hat{p}, \hat{n})$	58.09%	40.4

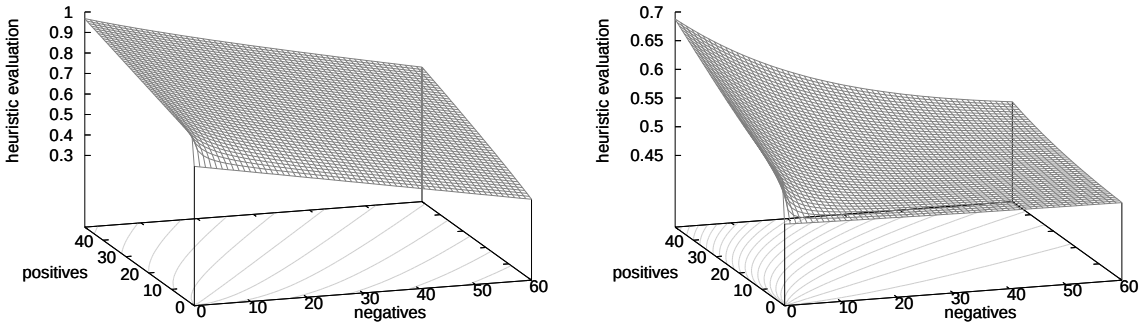
5.5 Predicting Other Heuristics

So far we focused on directly predicting the out-of-sample precision of a rule, assuming that this would be good heuristic for learning a rule set (cf. Section 3). 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 needed experiments, we decided to directly predict the number of covered positive ($\hat{p}$) and negative ($\hat{n}$) examples. We then 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 4 compares the performance of various heuristics with measured and predicted coverage values on the 30 test sets. In general, the results are disappointing. For three of the five heuristics, no significant change could be observed, but for *Weighted Relative Accuracy* and the *Correlation* heuristic, the performance degrades substantially.

A rather surprising observation is the 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



(a) Linear Regression (b) Neural Network

Figure 3: Isometrics of the functions (final rule precision)

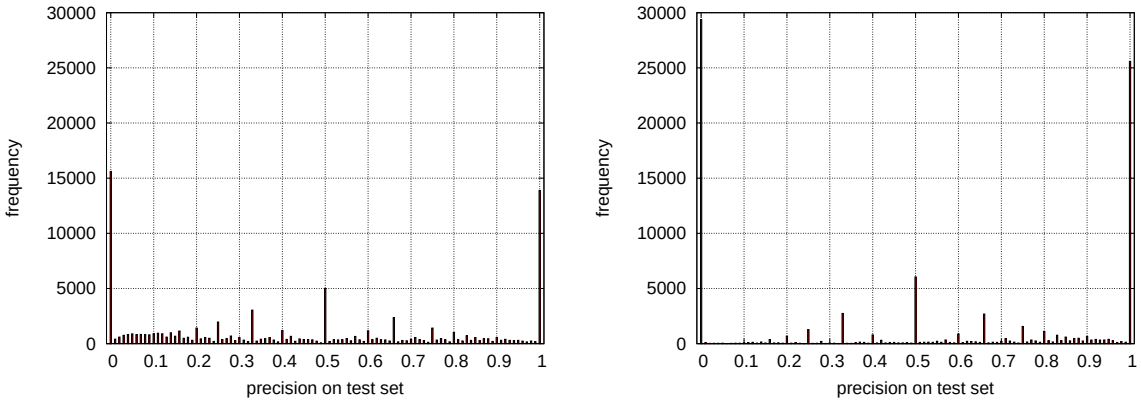
the predicted values reveals that on the one hand both regression methods predict negative coverages and that on the other hand for the region of low coverages (which is the important one) too optimistic values are predicted (for both the positive and the negative coverage). The acceptable performance 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).

Table 4 shows the results for predictions made by the neural network which performs best so far. Interestingly, the same network which was learned on the two meta data sets that include the length, performs consistently better, but the differences are not significant. All other neural networks and the linear regression perform disappointingly due to the above mentioned problems in the predictions.

5.6 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 Section 4.1, 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, in (Burges, 2006) it was 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. Thus, the new meta data set contains only 77,240 examples in total. For us, this seems to be the best way to handle the predecessors because if we want to evaluate them we could only use their immediate precision. But in the current approach we want to use the precision of the final rule, as described above.



(a) immediate precision

(b) final rule precision

Figure 4: Constitution of the meta data

A graphical interpretation of two sample sets can be found in Fig. 4 where, for a given precision, the number of instances were counted. If a candidate rule receives the same evaluation as its final rule (shown in Fig. 4 (b)), the frequency of the worst and the best evaluation increases. Additionally, there are also more rules with a precision of 0.5 too, which are mostly rules that cover a single positive and a single negative example.

All of these large rules often receive perfect training set precision but then are evaluated on the current sample of the test set[3]. On this part of the data, they often do not cover any positive example (i.e., their precision is 0) or no negative example (i.e., their precision is 1). These rules form the majority in the meta data which is preferable because they have the greatest variance among all rules. If a rule covers many examples in the training set, its precision will not differ significantly by that obtained on the test set. The more examples are covered in the training set, the lower is the probability that the rule will cover an entirely different number of examples on the test set. If, for example, a rule tests only one attribute and covers n examples on the training set, the probability that this rule will cover a significantly other number than n is small.

Table 5 shows the accuracies of the two heuristics that were learned in this setting, one with a linear regression, and one with a neural network with a single node in the hidden layer. In particular the neural network outperformed the original setting (cf. Table 2).

We also include the results of the 5 standard heuristics that were used to create the meta data. The induced heuristics outperform all of the standard heuristics. The Linear Regression was trained on the meta data set that only contains the 4 most important features which yield the best model. In terms of theory complexity it seems that about 50 conditions in average are necessary to obtain an accurate classifier. Weighted relative accuracy, for example, learns simpler theories (as observed in (Todorovski et al., 2000)), but seems to over-generalize. The neural network classifier performs best with the third-smallest theory.

Figure 3 shows the 3d-surfaces and the isometrics of the two learned heuristics. In comparison to Figure 2, it seems that their curvature towards the N-axis is considerably

3. As shown in Figure 1 the examples covered by the rule were removed from the test set too.

Table 5: Comparison of the induced heuristics with standard ones

heuristic	Accuracy	# conditions
Neural Network	78.37 %	53.97
Linear Regression	77.95 %	95.63
Correlation	77.57 %	47.50
Laplace	76.89 %	118.83
Precision	76.22 %	129.17
WRAcc	75.80 %	12.13
Accuracy	75.60 %	104.77

less steep, which is particularly visible at the points near the P-axis. However, the general shape of the curves seems to remain approximately the same.

6. Conclusion

The most important result of this work is that we have shown that a rule learning heuristic can be learned that outperforms standard heuristics in terms of predictive accuracy on a collection of databases that were not used in the meta-learning phase. Our first results, with used a few obvious features to predict the out-of-sample precision of the current rule, were already en par with the correlation heuristic, which performed best in our experiments. Subsequently, we tried to modify several parameters of this basic setup with mixed results. In particular, predicting the positive and negative coverage of a rule on a test set, and using these predicted coverage values inside the heuristics did not prove to be successful. Also, more complex neural network architectures did not seem to be important, linear regression and neural networks with a single node in the hidden layer performed best. On the other hand, a key result of this work is that evaluating a candidate rule by its potential of being refined into a good final rule works better for learning appropriate heuristics (both the linear regression and the neural network are more precise if they are learned on this type of data).

A visualization of the learned heuristics in coverage space gave some insight into the general functionalities of the learned heuristics. In comparison to heuristics with linear isometrics (such as precision, weighted relative accuracy, and the m-estimate that trades off between those two), the learned heuristics have non-linear isometrics that implement a particularly strong bias towards rules with a low coverage on negative examples. This makes sense for heuristics that will be used in a covering loop, because incompleteness (not covering all positive examples) can be compensated by subsequent rules, whereas inconsistency (covering too many negative examples) cannot. Correlation, the standard heuristic that performed best in our experiments, implements a similar bias (Fürnkranz & Flach, 2005). Thus, the results of this paper also contribute to our understanding of the desirable behavior of rule-learning heuristics.

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 in (Scheffer, 2001; Mozina, Demsar, Zabkar, & Bratko, 2006). In

other works, it has been addressed empirically. For example (Vapnik, Levin, & Cun, 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.

A promising direction for further research is to focus more strongly on the properties of the meta data. An interesting idea is to divide the learning problem into separate smaller problems by learning different models on coverage intervals. Thus, these models could be learned for rules which cover few, medium and many examples. Then, depending on the coverage of the current rule, the corresponding model can be used. Another direction is to focus more on the regression methods. Hence, a parameter optimization of the neural network or the usage of a Support Vector Machine will eventually yield to better results.

References

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

Burges, S. (2006). Meta-Lernen einer Evaluierungs-Funktion für einen Regel-Lerner.. Master Thesis.

Fürnkranz, J. (1994). FOSSIL: A Robust Relational Learner. *Lecture Notes in Computer Science*, *784*, 122–137.

Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, *13*(1), 3–54.

Fürnkranz, J. (2004). Modeling rule precision.. In *LWA*, pp. 147–154.

Fürnkranz, J., & Flach, P. A. (2003). An Analysis of Rule Evaluation Metrics. In *Proceedings 20th International Conference on Machine Learning (ICML'03)*, pp. 202–209. AAAI Press.

Fürnkranz, J., & Flach, P. A. (2005). ROC 'n' Rule Learning - Towards a Better Understanding of Covering Algorithms. *Machine Learning*, *58*(1), 39–77.

Janssen, F., & Fürnkranz, J. (2006). On trading off consistency and coverage in inductive rule learning.. In *LWA*, pp. 306–313.

Lavrač, N., Flach, P., & Zupan, B. (1999). Rule evaluation measures: A unifying view. In Džeroski, S., & Flach, P. (Eds.), *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pp. 174–185. Springer-Verlag.

Mozina, M., Demsar, 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. (1995). Inverse entailment and progol. *New Generation Comput.*, *13*(3&4), 245–286.

Newman, D., Blake, C., Hettich, S., & Merz, C. (1998). UCI Repository of Machine Learning databases..

Quinlan, J. (1996). Learning First-Order Definitions of Functions. *Journal of Artificial Intelligence Research*, *5*, 139–161.

Scheffer, T. (2001). Finding association rules that trade support optimally against confidence.. In *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001*, pp. 424–435.

Tan, P.-N., Kumar, V., & Srivastava, J. (2002). Selecting the right interestingness measure for association patterns.. In *KDD*, pp. 32–41.

Todorovski, L., Flach, P., & Lavrac, N. (2000). Predictive performance of weighted relative accuracy. In Zighed, D. A., Komorowski, J., & Zytkow, J. (Eds.), *4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD2000)*, pp. 255–264. Springer-Verlag.

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 edition). Morgan Kaufmann Publishers.