Modeling Rule Precision

Johannes Fürnkranz

TU Darmstadt, Knowledge Engineering Group
Hochschulstraße 10, D-64289 Darmstadt, Germany
fuernkranz@informatik.tu-darmstadt.de

Abstract. This paper reports first results of an empirical study of the precision of classification rules on an independent test set. We generated a large number of rules using a general covering algorithm and recorded their coverage on training and test sets. These meta data are briefly presented and analyzed with respect to their variance among different domains and search heuristics. The main part of the paper describes experiments that aimed at modeling the precision of the learned rules on the test set in dependence of their coverage on the training set. To this end, we trained a neural network as an evaluation function for a rule learner, and learned optimal parameter settings for the m-heuristic and the generalized m-heuristic. These settings are optimal in the sense that they minimize the squared error of predicting the test set precision with training set coverage of positive and negative examples.

1 Introduction

It is one of the fundamental facts of inductive learning that error estimates that are measured on the training set (*resubstitution estimates* or *empirical error*) are typically too optimistic. While there has been quite some work on theoretical error bounds on the true error, we are not aware of an empirical study that investigates this relation in practice. The particular focus of this work are error estimates of single rules inside a covering algorithm. Rules are particularly interesting to study because they differ not only in prediction quality but also in coverage. It is a largely an open question whether a good rule-based theory should consist of rules that cover many examples at the expense of a certain number of misclassifications or whether one should prefer rules that cover only few examples, but appear to be more precise. On the one hand, there is evidence that in many learning problems, large parts of the instance space need to be covered with small rules, but on the other hand, such rules are also known to be particularly unreliable. This is known as the *small disjuncts problem* (Holte et al., 1989).

In this paper, we introduce an experimental framework for studying rule prediction errors. We use a conventional covering or separate-and-conquer rule learning algorithm, described in detail in Section 2, to generate a large number of rules for a variety of benchmark datasets. The search heuristic of this algorithm is a parameter, so that we can implement several different strategies for trading off training set precision and coverage. This should guarantee a large variety in the type of generated rules. For all learned rules, we record the number of positive and negative examples that they cover on training and test sets of equal size. The details of the meta data generation are described in Section 3.

In Section 4, we show parts of the obtained meta data and provide a brief analysis. In particular, we will have a closer look at what we call *hermit rules*, i.e., rules that cover only a single positive and no negative training examples. This group of rules is by far the most frequent group encountered in our experiments. Our results show that the quality of hermit rules does not only depend on the domain but also on the heuristic used for searching the hypothesis space, which is a particularly interesting observation in the light of the results of (Holte et al., 1989).

Finally, in Section 5, we describe first experiments that use these meta data for modeling rule prediction errors and for meta-learning new heuristics. In particular, we trained a neural network evaluation function, and computed parameter settings for the m-estimat and the generalized m-estimate that are optimal in the sense that they minimize the squared error of the test set precision based on training set coverage. The results are quite reasonable (the fitted values do indeed improve performance over the base heuristics) albeit not ground-shaking. We also observe some evidence that the *a priori* probability of the positive class is an important parameter for rule learning heuristics. This, and other avenues for future research, are discussed in the concluding Section 6.

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 and Frank, 2000). Both the covering algorithm and the top-down refinement inside the covering loop are fairly standard, but covering algorithms often differ in details, so we believe it is worth-while to specify exactly how we proceeded.

Figure 1 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 routine REFINEMENTS returns the set of all candidate refinements that can been obtained by adding a single condition to the body of a rule. Conditions are either tests for equality with a specific value of a discrete attribute, or an inequality with some value of a continuous attribute (the value has to occur in the training set). All candidate refinements are evaluated with a heuristic, and the best one is selected. If no further refinements are possible, the search stops and the best rule encountered during the search is returned.

Thus, the algorithm works like CN2 (Clark and Niblett, 1989), but differs from Foil (Quinlan, 1990), which forms the basis of many rule learning algorithms, most notably Ripper (Cohen, 1995). Foil-based algorithms do not evaluate refinements on an absolute scale, but relative to their respective predecessors. Hence, the evaluation of two rules with different predecessors are not directly comparable. For this reason, Foil-like algorithm 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. For the type of algorithm shown in Figure 1 this is not so crucial, because the rule returned is not necessarily the last rule searched, but the rule with the highest evaluation encountered during the search.

```
function 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
```

Fig. 1. Greedy top-down search for the best rule.

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. This observation was first made by Clark and Boswell (1991) for the CN2 rule learning algorithm.

Our main goal is to study the test set performance of individual rules, and not so much to learn a good theory. Therefore, the evaluation of possibly overfitting rules (e.g., hermit rules) is very important to us. As a consequence, we chose not to implement any filtering heuristics which would prune those rules away. To ensure some variety in the size of the learned rules by using evaluation heuristics with very different biases (as will be explained below), some of which have a tendency to learn very general rules, while others are clearly prone to overfitting.

Our implementation of the algorithm made use of a few optimizations that are not shown in Figure 1. 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 ex-

```
function 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 pos than negs
    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
```

Fig. 2. Covering algorithm

amples and none of the remaining negative examples—has a lower evaluation than the current best rule.

The algorithm for finding a single rule is used in an outer covering loop (see Figure 2) which repeatedly learns one rule, 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).

3 Meta Data Generation

We used the simple rule learner described above for obtaining various characteristics of learned rules. For the work described here, we were interested in comparing the estimates for the precision of a rule on the training data to the estimate on an independent test set. Thus, for each learned rule, we recorded the numbers of covered positive and negative examples on both the training and an independent test set.

We recorded these statistics not only for *final rules*—those rules that would be used in the final theory—but also for all their ancestors, i.e., for all *incomplete rules* that were eventually refined into a final rule. These can be simply obtained by deleting the final conditions of each rule. The main motivation for this step is that we want to have complete information on each path in the refinement graph that yields a final rule. Figure 3 shows the meta data generation algorithm in pseudo-code.

In order to collect statistics under a fairly broad set of conditions, we varied the following dimensions:

```
procedure GENERATERULES(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
    TrainP ← |COVERED(Rule,POSITIVE(TrainSet))|
    TrainN ← |COVERED(Rule,NEGATIVE(TrainSet))|
    TestP ← |COVERED(Rule,POSITIVE(TestSet))|
    TestN ← |COVERED(Rule,NEGATIVE(TestSet))|
    print Pred,Rule,TrainP,TrainN,TestP,TestN

    Pred ← REMOVELASTCONDITION(Pred)
  until Pred = null

  # remove covered training and test examples
  TrainSet ← TrainSet \ COVERED(Rule,TrainSet)
  TestSet ← TestSet \ COVERED(Rule,TestSet)
```

Fig. 3. Covering algorithm for generating and evaluating rules

Datasets: We used 27 datasets with varying characteristics from the UCI repository. These datasets were selected because of their availability and moderate size. We did not include larger datasets (such as *shuttle*) because the region of interest (as we will see) is the region of rules with low coverage. Such rules can be equally well found in smaller datasets, which we favored because of efficiency.[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 estimates. We performed five iterations with different random seeds, as suggested by Dietterich (1998). Note, however, that our purpose was not accuracy estimation of the entire theory, but for individual rules. Therefore, we collected statistics for all rules of all folds.

[1] Efficiency was important because heuristics like *Precision* may end up in learning one rule for each example in the training set, which makes experimentation in large domains very costly.

Table 1. Search heuristics used in this study. p and n are the number of covered among a total of P and N positive and negative examples.

heuristic	formula
precision	$\frac{p}{p+n} \sim \frac{p-n}{p+n}$
Laplace	$\frac{p+1}{p+n+2}$
accuracy	$\frac{p+(N-n)}{P+N} \sim p-n$
weighted rel. acc.	$\frac{p+n}{P+N}(\frac{p}{p+n} - \frac{P}{P+N}) \sim \frac{p}{P} - \frac{n}{N}$
correlation	$\frac{p(N-n)-(P-p)n}{\sqrt{PN(p+n)(P-p+N-n)}}$

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

Heuristics: Finally, we ran the rule learner five times on each binary dataset, each time using a different search heuristic. We used the five heuristics shown in Table 1. The first four form a representative selection of search heuristics with linear isometrics (Fürnkranz and Flach, 2003), while the correlation heuristic (Fürnkranz, 1994) 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* tend to prefer possibly complex rules with high precision on the training set. Fürnkranz and Flach (2003) have pointed out that these heuristics are often equivalent to other heuristics (Table 1 shows a few examples for equivalences).

In total we learned 5409 theories with a total of 48,603 rules. For all these rules and their ancestors we recorded the total number of covered positive and negative examples in the test set, resulting in statistics for a total of 114,375 rules. 13,399 rules did not cover any examples on the test set and were ignored. 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 test example). Ignoring them seemed to be the most reasonable option for our purposes. The large majority of these ignored rules (9,806 rules) covered only a single positive and no negative examples on the training set. A total of 100,976 rules remained for the analysis.

Each rule is evaluated in the context of all previously learned rules, i.e., all examples covered by previous rules are removed from the dataset. Thus, later rules in a theory are learned from a smaller dataset than the first rules in the theory. This procedure was also mirrored in the test set. In other words, we assumed a decision list learning scenario, where an example is classified with the prediction of the first rule that fires on the example. Thus, rule n only receives examples (from both training or test sets) that are not covered by rules $1 \ldots n-1$.

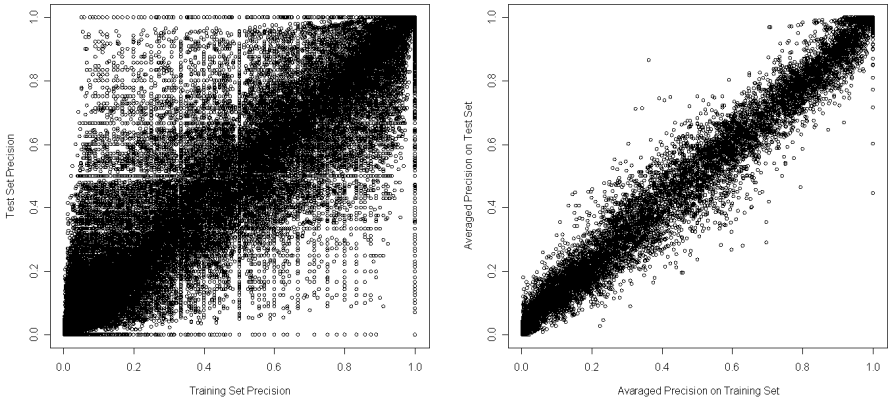

Fig. 4. Precision on training set vs. precision on test set. The left graph shows the raw data (one point for each rule), while the right graphs shows the plot for averages (one point for each set of rules that cover the same number of positive and negative examples).

4 The Meta Data

In this section, we analyze the meta data that was produced as described in the previous section. What interests us most is the relation between the precision of a rule on the training set and its precision on the test set, which we assume is a better estimate of the rule's true quality. Note that both estimates are derived from data samples of the same size (due to the 2-fold cross-validation) so that the statistical quality of both estimates should be comparable.

The left graph of Figure 4 shows a scatter plot of training set precision vs. test set precision. Obviously, the values are denser near the diagonal, indicating a strong correlation between the two. The closer we get towards the points $(0, 1)$ or $(1, 0)$, the more scattered are the data points. A few lines (e.g., at 1, 1/2, 1/3 and 3/4) are more densely populated than others. These are due to a large number of low coverage rules, which tend to have a fairly large variance. For example, there are many rules with perfect training precision, most of which are rules that cover exactly one training example. Similarly, most of the rules with perfect test precision cover exactly one example on the test set (but possibly more on the training set). We will analyze rules covering one positive and no negative examples further below (Section 4.1).

The right half of Figure 4 shows the same graph for the case where precision values for rules with identical coverage have been averaged (i.e., all rules that cover exactly one positive example on the training set are shown as a single point). Obviously, this makes the correlation between training and test precision more apparent.

More interesting is a tabulation of the data according to the absolute number of covered positive and negative training examples. Table 2 shows these data. Each entry in the upper half shows an average of the test-set precision estimates of all rules that cover the indexed number of positive and negative training examples. The lower half shows

Table 2. Average precision on the test set (top) and number (bottom) of rules with a given coverage on the training set. The rows are the number of covered positive examples, the columns the number of covered negative training examples.

p/n	0	1	2	3	5	7	10	15	20	30	50	70	100
1	0.446	0.350	0.324	0.280	0.239	0.218	0.207	0.167	0.115	0.073	0.053	0.084	0.065
2	0.601	0.456	0.393	0.367	0.308	0.309	0.240	0.217	0.182	0.120	0.077	0.043	0.042
3	0.715	0.542	0.488	0.463	0.387	0.323	0.290	0.272	0.213	0.102	0.059	0.048	0.085
5	0.808	0.652	0.606	0.568	0.485	0.475	0.455	0.343	0.249	0.153	0.092	0.071	0.062
7	0.850	0.728	0.696	0.639	0.579	0.540	0.443	0.403	0.288	0.347	0.199	0.126	—
10	0.898	0.786	0.723	0.695	0.625	0.571	0.534	0.490	0.392	0.333	0.157	0.150	0.173
15	0.943	0.849	0.794	0.754	0.703	0.629	0.552	0.559	0.549	0.513	—	0.197	0.172
20	0.952	0.865	0.845	0.830	0.750	0.747	0.632	0.693	0.582	0.538	—	0.192	—
30	0.981	0.859	0.860	0.766	0.727	0.673	0.629	0.749	0.583	0.573	0.165	0.381	0.233
50	0.991	0.969	0.900	0.911	0.842	0.828	0.982	0.802	0.783	—	0.381	—	—
70	0.973	0.961	0.973	0.993	0.976	0.884	0.868	0.845	—	—	—	0.480	—
100	1.000	—	—	0.971	—	—	0.928	0.889	0.821	—	0.737	—	—
1	15877	3719	2112	1494	899	634	432	264	198	110	51	22	5
2	5037	1520	890	615	398	268	212	143	66	38	16	9	2
3	2645	879	531	413	268	175	149	85	50	22	9	7	3
5	1172	358	275	186	125	113	74	71	31	8	8	4	4
7	658	220	170	124	102	64	48	35	23	11	6	4	0
10	359	126	90	82	48	50	31	37	21	7	2	2	3
15	167	43	49	33	36	25	18	12	12	3	0	3	2
20	126	33	25	26	14	11	11	9	3	1	0	1	0
30	59	13	12	12	15	10	6	11	3	5	1	1	1
50	10	3	2	4	2	2	1	5	3	0	1	0	0
70	2	2	2	2	1	3	3	2	0	0	0	1	0
100	3	0	0	1	0	0	2	1	1	0	1	0	0

the number of rules that have this particular coverage in the meta data. For example, a rule covering one positive and no negative examples on the training set has an estimated true precision of 0.446. This estimate is the average of the test precision of 15,877 rules with that coverage in the meta data.

Overall, the numbers appear to be fairly consistent, with fairly smooth growth functions along the dimensions. For example, the estimates for rules that cover no negative examples approach 1 fairly smoothly (cf. also Figure 5). It is interesting to look at the relative size of different values. For example, the point (1,0) has a somewhat higher estimated precision than (7,15) but a lower estimate than (10,15). We also note that the numbers in regions with a low number of rules become less reliable and "inconsistent" (e.g., points (1,50) and (1,100), or the values on the diagonal which should converge towards 0.5 on the test set).

4.1 Hermit Rules

Next, we will have a closer look at the most frequent data point, the 15,877 rules covering exactly one positive and no negative examples on the training data. We will call

Table 3. Average test precision of hermit rules, tabulated by dataset and search heuristic. Also shown are the number of such rules for each combination.

	Precision		Laplace		Accuracy		WRA		Correlation	
anneal	0.848	119	0.803	32	0.653	72	0.500	4	0.812	20
audiology	0.491	338	0.419	225	0.439	220	0.559	49	0.502	110
breast-cancer	0.570	307	0.541	202	0.338	102	0.000	4	0.354	70
cleveland-heart-disease	0.554	328	0.404	128	0.467	98	0.479	8	0.427	42
contact-lenses	0.302	32	0.197	25	0.189	22	0.317	10	0.169	18
credit	0.558	351	0.321	166	0.284	139	0.000	6	0.231	42
glass	0.443	335	0.271	172	0.336	217	0.205	11	0.266	62
glass2	0.612	220	0.421	77	0.299	66	0.491	9	0.410	16
hepatitis	0.611	154	0.343	67	0.301	49	0.416	8	0.389	25
horse-colic	0.577	407	0.512	162	0.327	122	0.222	9	0.360	46
hypothyroid	0.731	368	0.376	80	0.448	94	0.036	7	0.229	35
iris	0.803	69	0.545	11	0.751	23	0.358	6	0.786	7
krkp	0.631	310	0.512	181	0.386	258	0.000	2	0.412	11
labor	0.744	57	0.700	25	0.727	11	0.750	7	0.795	11
lymphography	0.579	130	0.536	60	0.517	62	0.700	8	0.518	24
monk1	0.543	77	0.627	36	0.570	32	0.700	10	0.625	19
monk2	0.463	183	0.438	135	0.377	65	0.425	12	0.412	59
monk3	0.582	57	0.519	43	0.304	19	0.500	2	0.426	9
mushroom	1.000	14	1.000	9	1.000	11	—-	0	1.000	7
sick-euthyroid	0.764	708	0.500	212	0.284	102	0.250	4	0.167	59
soybean	0.431	332	0.334	217	0.321	261	0.475	4	0.467	52
tic-tac-toe	0.496	71	0.475	18	0.495	184	0.000	1	0.292	6
titanic	—-	0	1.000	1	1.000	2	—-	0	—-	0
vote	0.578	86	0.341	46	0.256	33	0.042	8	0.322	15
vote-1	0.573	171	0.427	93	0.300	57	0.292	8	0.415	28
vowel	0.369	1900	0.266	1036	0.274	1594	0.284	17	0.298	170
wine	0.744	130	0.585	31	0.398	18	0.056	6	0.591	17
average	0.578	268.7	0.497	129.3	0.446	145.7	0.298	8.1	0.432	36.3

such rules *hermit rules*. Table 3 shows the average test set precision of hermit rules on various datasets and heuristics.

Not surprisingly, we can observe differences in the quality of hermit rules between different domains. For example, all hermit rules that were learned in the *mushroom* domain turned out to be 100% correct on the test set whereas all six hermit rules learned by WRA in the *credit* domain had 0% precision on the test set. The fact that the general trend whether the values are high or low is consistent among the various heuristics, suggests that the differences cannot solely be attributed to variance, but that they depend on domain characteristics. A reasonable explanation for the variance between domains is that the performance of a bad hermit rule (e.g., a rule that covers a single noisy example) is likely to depend on the prior probability of the positive examples in the domain, because if the rule is based on a noisy example, on average, the performance of such rules on the test set can be expected to be close to that of a random rule.

However, more surprisingly, the quality of hermit rules also varies considerably with the heuristic that was used for finding them. These differences are interesting because, of course, the heuristics can not discriminate between a "good" and a "bad" hermit rule, they only differ in the way they evaluate a hermit rule compared to other rules. A possible explanation could be that heuristics like *WRA*, which tend to learn more general rules (Todorovski et al., 2000), will only cover isolated and therefore unreliable points with hermit rules; fairly dense and uniform areas will be covered by larger rules. *Precision*, on the other hand, with its strong bias for pure rules, is more likely to cover even parts of fairly uniform areas with hermit rules. The more uniform a region is, the more likely it is that the test examples that are covered by a hermit rule are of the same class as the single example that yielded the rule. Thus, these rules can be expected to be of fairly high quality.

It is also interesting to note the connection to the *small disjuncts problem* (Holte et al., 1989), where it was shown that even though small rules tend to be unreliable, they are nevertheless necessary for maintaining a certain level of classification accuracy. Our experiments suggest that the quality of small rules may crucially depend on the neighborhood of the covered examples in example space. It might be interesting to re-evaluate the small disjuncts problem in this light.

5 Meta Learning

In this section we will have a closer look at the distribution of the test set precision and try to fit functions to their values. We will first look at the behavior for fixed values of p or n (Section 5.1), and then try to fit parameterized search heuristics to the data (Section 5.2).

5.1 Fixed values of p and n

Figure 5 shows a plot of the distribution of the test precision over the p-values for fixed values of $n = 0, 5, 10$ (top), and their distribution over the n-values for fixed values of $p = 1, 5, 10$, along with a curve that shows the training set precision (solid line). The solid line shows the overly optimistic precision estimates that are obtained from the training set, in the case of $n = 0$ a constant 1. The real values seem to follow a general power law distribution (Newell and Rosenbloom, 1981). Using R's `nls` function, which fits parameters in non-linear functions using an iterative Gauss-Newton algorithm (Venables and Ripley, 2002), we obtained the best fit with the functions

$$prec_{n=0} = 1.002 - 1.177p^{-1.033} \qquad prec_{p=1} = -0.033 + 0.504n^{-0.375}$$

with residual errors of 0.0235 and 0.1265 respectively.

With increasing values of p or n, the fitted functions more and more approximate the curve for the training set precision $prec = p/(p+n)$. At the same time, the residual errors of the fit increase because the observed test set precision estimates originate from smaller number of rules and vary much more around the optimal fit. Details on these results can be found in (Fürnkranz, 2003).

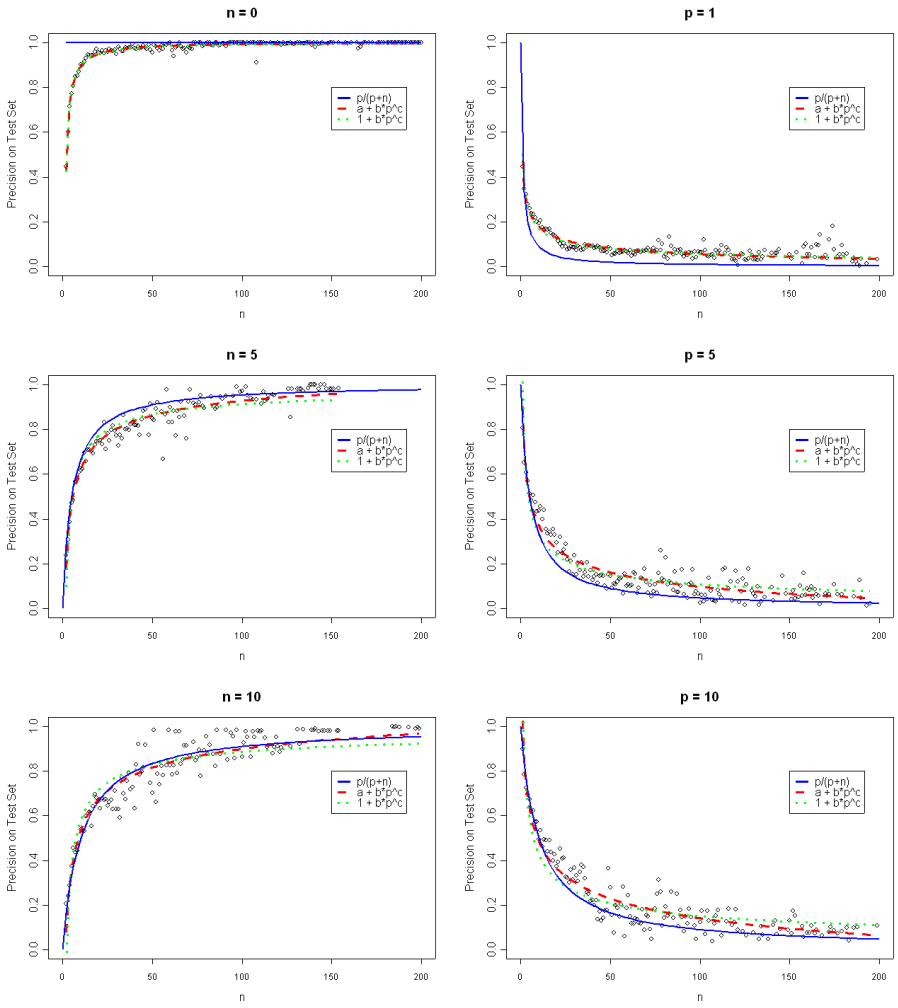

Fig. 5. Scatter-plots, the curve $p/(p+n)$, and fitted curves for the test set precision of rules that cover no negative examples ($n = 0$).

5.2 Fitting Search Heuristics

The main goal in the previous section was descriptive, we wanted to understand the general shape of the error surface on the test set. In this section, we will try to fit 2-dimensional functions to this surface, with the explicit goal of using them as a search heuristic inside the rule learner. The main motivation for this approach is that we want to evaluate candidate rules with better estimates of their true precision. We fitted the parameters of the following three types of heuristics:

Table 4. Accuracy and number of learned rules for the five basic and three learned heuristics on 10 new datasets. For comparison, we also show the result of JRip without Pruning (-P) and JRip, and the average results of the algorithms on the 27 datasets that were used for training.

	Prec	Lap	Acc	WRA	Corr	NNet	MEst	GenM	JRip -P	JRip
average (27 old)	81.97	81.84	81.90	82.48	83.09	82.55	82.74	83.00	83.95	84.71
balance-scale	73.44	73.12	68.80	66.88	77.76	71.84	72.32	72.80	80.32	81.28
breast-w	94.85	94.85	95.28	94.28	95.57	95.14	94.56	95.14	93.71	95.14
credit-g	69.10	70.00	67.00	72.50	69.60	67.70	68.10	69.20	73.30	70.80
diabetes	68.23	69.66	69.27	71.88	69.01	70.31	71.88	68.75	72.92	74.22
ionosphere	93.45	94.30	89.46	89.74	88.03	94.02	93.73	94.02	90.60	88.60
primary-tumor	33.04	32.74	29.50	35.40	35.40	33.63	34.81	33.33	39.23	38.94
segment	91.39	90.61	88.10	92.29	94.94	91.64	91.77	91.17	95.76	95.11
sonar	62.02	63.46	68.75	67.79	73.08	66.83	65.87	67.31	77.40	76.44
vehicle	69.39	67.14	62.65	60.52	68.44	65.48	71.63	67.49	67.02	68.68
zoo	84.16	85.15	90.10	92.08	90.10	89.11	90.10	90.10	87.13	86.14
average (10 new)	73.91	74.10	72.89	74.34	76.19	74.57	75.48	74.93	77.74	77.53
avg. # rules (27 old)	40.41	36.93	32.56	4.74	13.63	30.22	30.81	30.85	14.11	8.11
avg. # rules (10 new)	83.20	78.30	86.50	4.30	21.20	64.20	67.60	68.40	18.50	9.80

- a neural network (fully connected with a five-node hidden layer, fitted using R's `nnet` procedure (Venables and Ripley, 2002))
- the m-heuristic (Cestnik, 1990), resulting in the function $\frac{p+1.6065*P/(P+N)}{p+n+1.6065}$ (fitted using R's `nls` procedure (Venables and Ripley, 2002))
- the generalized m-heuristic, which re-interprets the prior probability in the m-heuristic as a cost parameter (Fürnkranz and Flach, 2003), resulting in $\frac{p+0.785}{p+n+2.7153}$

The residual sum-of-squares showed the best fit for the m-heuristic ($rss = 7842.37$), followed by the neural network ($rss = 7897.1$) and the generalized m-heuristic ($rss = 8029.34$). One lesson that we take from the difference between the fit of the versions of the m-heuristic is that the prior probability seems to be an important parameter for fitting a good evaluation function.

Table 4 shows the average accuracy (estimated by a 10-fold stratified cross-validation) and the number of learned rules for all eight heuristics[2] on the 27 data sets that were used for generating the meta data, as well as detailed results for 10 new datasets. As an independent benchmark, we also added the results of JRip, Weka's re-implementation of Ripper (Cohen, 1995), in two versions, without and with pruning. Table 5 shows the p-value of a paired t-test, and the number of wins and losses for each combination of a base heuristic with a meta-learned heuristic.

Among the heuristics with linear isometrics (all except the neural network and *Correlation*), the trained m-estimates clearly outperform *Precision*, *Laplace*, and *Accuracy*, in all but one case at the 1% significance level. Weighted relative accuracy is insignifi-

[2] The neural network was implemented via a look-up table of the average prediction values of 10 different networks for all combinations of values $n \leq 50$ and $p \leq 50$. Precision was used for all larger values.

Table 5. Significance level of a paired t-test and number of wins and losses of pairwise comparisons between the base heuristics and the meta-learned heuristics.

	NNet	MEst	GenM
Pre	0.929 (10/23)	0.996 (9/25)	0.9996 (7/26)
Lap	0.913 (10/23)	0.991 (12/21)	0.999 (10/22)
Acc	0.939 (13/21)	0.985 (13/22)	0.996 (12/23)
WRA	0.541 (20/15)	0.679 (16/19)	0.695 (16/19)
Corr	0.178 (21/15)	0.291 (20/15)	0.321 (20/15)

cantly behind in accuracy, but achieves this performance with a much lower number of rules. This can be seen as more evidence for the importance of the use of prior probabilities (WRA normalizes accuracy with the prior probability) that we had already presumed above from the comparison of the quality of fits.

Overall, the correlation heuristic outperformed all other heuristics that we tried. Quite possibly, its good performance is due to the non-linear shape of its isometrics and to the fact that, like WRA, it implicitly normalizes for the prior probability of the problem.

The performance of the neural network is somewhat disappointing. Although it is the most expressive model class (the only one that could be trained to fit non-linear isometrics), the net did not surpass the results of its linear competitors, not even at a significance level of 5%. Overfitting could be one cause, the above-mentioned absence of the prior probability as an additional input to the network another.

Nevertheless, it is interesting to see how the network fitted the data. Figure 6 shows the surface of the learned evlauation function. Note that the steep non-linear shape for low levels of N and P gradually shifts towards an almost linear shape. This is not surprising, as the bias of the training set precision can be expected to be much lower for rules with high coverage than for rules with low coverage, because there are better chances that a rule can fit a smaller sample by chance.

Figure 7 shows the isometrics of the learned neural network. It is quite obvious that the shape is very similar to the shape of precision, which would be lines rotating around the angle $(0, 0)$. However, while the lines become increasingly straight the farther they move away from the origin, they are quite non-linear near the origin. In these regions, it might make a difference whether a rule is evaluated with precision on the training set or with the predicted test set precision. Moreover, the isometrics do not meet in the origin, but seem to rotate around some point below it. This is characteristic of the m-estimate, and may partly explain the good performance of these heuristics.

In general, our results are on average somewhat below those of JRip, although there are numerous exceptions. This difference could be due to a variety of reasons, one of them being minor differences in the implementation. For example, we added the constraint that a rule has to cover more positive than negative examples relatively late in our experiments, but it had a great influence on the performance (in particular for the WRA heuristic). Possibly, there are one or two more subtle differences that have an impact on the performance of the two algorithms. Another reason could be the difference in the internal rule learner: as noted above, the implementation of JRip returns the last refinement, whereas our implementation returns the rule with the highest evaluation en-

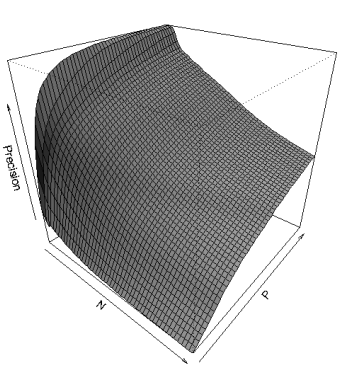

Fig. 6. Surface of a neural-net fit to the evaluation data

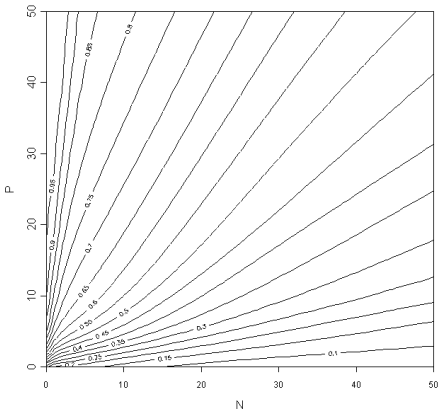

Fig. 7. Isometrics of a neural-net fit to the evaluation data

countered in the refinement process. We consider it still to be an open question, which approach is preferable. A somewhat unexpected side result of our experiments is that the no pruning version of JRip often outperforms its pruning counter-part (17 wins vs. 18 losses, with a p-value of 0.86). Thus, it can be assumed that the lack of a pruning option does not necessarily hamper the performance of our simple implementation of the separate-and-conquer algorithm on this selection of datasets.

6 Discussion

In our view, it is still a largely open question what search heuristics should be used for rule learning. In fact, it is not even clear what an ideal search heuristic should measure. Current heuristics use the performance of an incomplete rule on the training data for evaluating its utility. This is suboptimal for at least two reasons:

- performance measures on the training data are optimistically biased
- incomplete rules should not be assessed by their own performance, but by their potential of being refined into a high-quality rule

The work reported in this paper has addressed the first problem by investigating the relation between training set and test set precision estimates of rules. While we know from theoretical bounds that training set precision estimates are optimistically biased, we believe that this is the first empirical study of this issue. We mainly focused on "correcting" training set precision estimates with a learned estimate of test set precision because, as was shown by Fürnkranz and Flach (2003), precision has the goal of optimizing the area under the ROC-curve under unknown costs, whereas other heuristics, such as accuracy or WRA, make an implicit or explicit assumption about misclassification costs for a single rule.

The second problem remains open. Commonly used search heuristics for rule learning do not differentiate between the evaluation of a final rule and the evaluation of an incomplete candidate rule. As noted above, the latter should not be evaluated with their ability to discriminate between positive and negative examples, but with their potential to be refined into a rule that has this ability. These are two different issues which are not separated by current rule learning algorithms. A possible way for solving this problem could be to adapt the meta-learning scenario laid out in this paper, and use a reinforcement learning approach for learning evaluation functions for incomplete rules.

A by-product of our work is an empirical comparison of five rule learning heuristics, among which the correlation heuristic clearly emerged best. In future work, we plan to consolidate these results with the inclusion of filtering and pruning heuristics, whose absence is likely to have a larger impact on heuristics that produce a larger number of hermit rules (or small disjuncts).

A frequently heard comment on our work is that we should take more features into consideration, such as features that characterize the domain (e.g., number of numeric/symbolic attributes, etc.) or the learned rule (e.g., length of the rule or position of the rule in the rule set). These are interesting and promising lines for further research in meta-learning, but they go far beyond our primary goal of understanding commonly used search heuristics. The motivation for our detour into meta-learning originates directly from previous work (Fürnkranz and Flach, 2003), in which we investigated the isometric landscape of commonly used rule evaluation metrics. Almost all of them use p and n as their sole parameters. The goal of our work is to search for the "optimal" search heuristic in this PN-space. However, we take it as one important result of our work that we have some indication that search heuristics which take the prior probability of the domain into account (such as weighted relative accuracy or the m-estimate) perform rather well, whereas the performance of meta-learned heuristics that did not use this parameter (such as the generalized m-estimate and the neural network) was disappointing. In future work, we plan to include this as a third parameter into our meta-learning experiments, thereby effectively moving operation into 3D-ROC space (Flach, 2003).

Finally, we need to pay increased attention to functions with non-linear isometrics (as witnessed by the good performance of the correlation heuristic). In particular, we hope to find a suitable function family that is able to model the non-linear isometrics of the test set precision function in an explicit form, thereby hopefully also improving upon the rather disappointing performance of the neural net-based search heuristic.

Acknowledgments

I would like to thank the anonymous reviewers of this and previous versions of this paper for many thoughtful comments that helped to improve this paper. Much of this work was supported by an *APART stipend* (no. 10814) of the Austrian Academy of Sciences, and by many people that make great software freely available. Weka, R, Perl and Cygwin were invaluable resources for this project.

References

B. Cestnik. Estimating probabilities: A crucial task in Machine Learning. In L. Aiello (ed.) *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, pp. 147–150, Stockholm, Sweden, 1990. Pitman.

P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session on Learning (EWSL-91)*, pp. 151–163, Porto, Portugal, 1991. Springer-Verlag.

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

W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell (eds.) *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pp. 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

P. A. Flach. The geometry of ROC space: Using ROC isometrics to understand machine learning metrics. In T. Fawcett and N. Mishra (eds.) *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 194–201, Washington, DC, 2003. AAAI Press.

J. Fürnkranz. FOSSIL: A robust relational learner. In F. Bergadano and L. De Raedt (eds.) *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*, pp. 122–137, Catania, Italy, 1994. Springer-Verlag.

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

J. Fürnkranz. Modeling rule precision. Technical Report OEFAI-TR-2003-35, Austrian Research Institute for Artificial Intelligence, Wien, Austria, 2003. URL `http://www.oefai.at/cgi-bin/tr-online?number+2003-35`.

J. Fürnkranz and P. Flach. An analysis of rule evaluation metrics. In T. Fawcett and N. Mishra (eds.) *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 202–209, Washington, DC, 2003. AAAI Press.

R. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, pp. 813–818, Detroit, MI, 1989. Morgan Kaufmann.

A. Newell and P. S. Rosenbloom. Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (ed.) *Cognitive Skills and Their Acquisition*, chapter 1, pp. 1–51. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1981.

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

L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In D. Zighed, J. Komorowski, and J. Zytkow (eds.) *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)*, pp. 255–264, Lyon, France, 2000. Springer-Verlag.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, fourth edition, 2002. URL `http://www.r-project.org/`.

I. H. Witten and E. Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2000. URL `http://www.cs.waikato.ac.nz/~ml/weka/`.