A Tight Integration of Pruning and Learning*

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria

E-mail: juffi@ai.univie.ac.at

Abstract

This paper outlines some problems that may occur with *Reduced Error Pruning* in rule learning algorithms. In particular we show that pruning complete theories is incompatible with the *separate-and-conquer* learning strategy that is commonly used in propositional and relational rule learning systems. As a solution we propose to integrate pruning into learning and examine two algorithms, one that prunes at the clause level and one that prunes at the literal level. Experiments show that these methods are not only much more efficient, but also able to achieve small gains in accuracy by solving the outlined problem.

Keywords: Rule Learning, Inductive Logic Programming, Pruning, Noise

OEFAI-TR-95-03

*An Extended Abstract of this paper appeared in the Proceedings of the ECML-95.

1 Introduction

Most rule learning algorithms deal with noise in the data during learning, i.e. they employ *pre-pruning*. In relational learning systems such as FOIL (Quinlan and Cameron-Jones 1993), *m*FOIL (Džeroski and Bratko 1992), or FOSSIL (Fürnkranz 1994b) pre-pruning is commonly used in the form of so-called *stopping criteria*. An alternative way for dealing with noise — *post-pruning* — is to first learn a theory that overfits the data and then prune this theory to an appropriate level of generality. The most common post-pruning algorithm, *Reduced Error Pruning (REP)*, has been adapted from propositional decision tree learning (Quinlan 1987) to relational rule learning (Brunk and Pazzani 1991). However, this adaptation has several shortcomings. In particular we will argue in section 2 that post-pruning complete theories is incompatible with the commonly used *separate-and-conquer* rule learning strategy. As a solution we propose to integrate pruning into learning and examine two algorithms, one that prunes at the clause level (section 3) and one that prunes at the literal level (section 4). In section 5 we will report some experiments in relational and propositional domains which show that these methods can improve the learning process in terms of speed and accuracy.

2 REP

Reduced Error Pruning (REP) was originally proposed in (Quinlan 1987) as a method for post-pruning decision trees. Pagallo and Haussler (1990) adapted it for learning decision lists. A version that can be used for relational learning was then introduced in (Brunk and Pazzani 1991).

The basic algorithm of this version is depicted in Fig. 1. After splitting the training set into a growing and a pruning set according to some user-specified ratio, a concept description that covers all of the positive and none of the negative examples of the growing set is learned with a separate-and-conquer rule learning algorithm like the propositional learner CN2 (Clark and Niblett 1989) or the relational learner FOIL (Quinlan 1990). This intermediate theory is then simplified by deleting literals and clauses until any further deletion would lead to a decrease of accuracy on the pruning set.

The major shortcomings of this straightforward adaptation of REP for rule learning are its inefficiency and its incompatibility with the separate-and-conquer search strategy that is commonly employed in propositional and relational rule learning algorithms. REP is very inefficient, because the overfitting theory it generates in its first pass can be much more complex than the final theory that is left after the post-pruning phase. A lot of work is wasted in learning and subsequently pruning superfluous literals and clauses. This argument has been formalized in (Cohen 1993), where it was shown that the growing phase of REP has a time

```
procedure REP(Examples, SplitRatio)

SplitExamples(SplitRatio, Examples, GrowingSet, PruningSet)
Theory = SeparateAndConquer(GrowingSet)
loop
    NewTheory = SimplifyTheory(Theory,PruningSet)
    if Accuracy(NewTheory,PruningSet) <
                Accuracy(Theory,PruningSet)
       exit loop
    Theory = NewTheory
return(Theory)
```

Figure 1: Reduced Error Pruning

complexity of $\Omega(n^2 \log n)$ and that its pruning phase has a time complexity of $\Omega(n^4)$ (where n is the size of the training set).

Fürnkranz and Widmer (1994) point out another problem with REP that is caused by the differences between the *divide-and-conquer* approach used for decision tree learning and the *separate-and-conquer* strategy commonly used for rule learning. The decision tree learning algorithms of the TDIDT family (Quinlan 1986) all use a *divide-and-conquer* strategy. After having selected an appropriate test for the root note of the decision tree, the algorithm divides training set into disjoint subsets, each one representing a possible outcome of the chosen test. The algorithm is then recursively applied to each of these sets independently.

On the other hand, greedy covering algorithms like Foil follow a *separate-and-conquer* approach. Here the learner constructs the theory rule by rule. After a rule is learned the algorithm separates all examples that are covered by this rule from the current set of training examples. Then the algorithm is recursively applied to find rules that explain the remaining examples.

Although the separate-and-conquer approach shares many similarities with the divide-and-conquer strategy, there is one important difference: Pruning of branches in a decision tree will never affect the neighboring branches, whereas pruning of literals of a rule will affect all subsequent rules. Figure 2 (a) illustrates how post-pruning in decision tree learning works. The right half of the initially grown tree covers the sets C and D of the training instances. When the pruning algorithm decides to prune these two leaves, their ancestor node becomes a leaf that now covers the examples $C \cup D$. The left branch of the decision tree is not influenced by this operation.

Pruning a literal from a clause on the other hand means that the clause is generalized, i.e. it will cover more positive instances along with some negative instances. Consequently those additional positive and negative instances should be removed from the training set so that they cannot influence the learning of

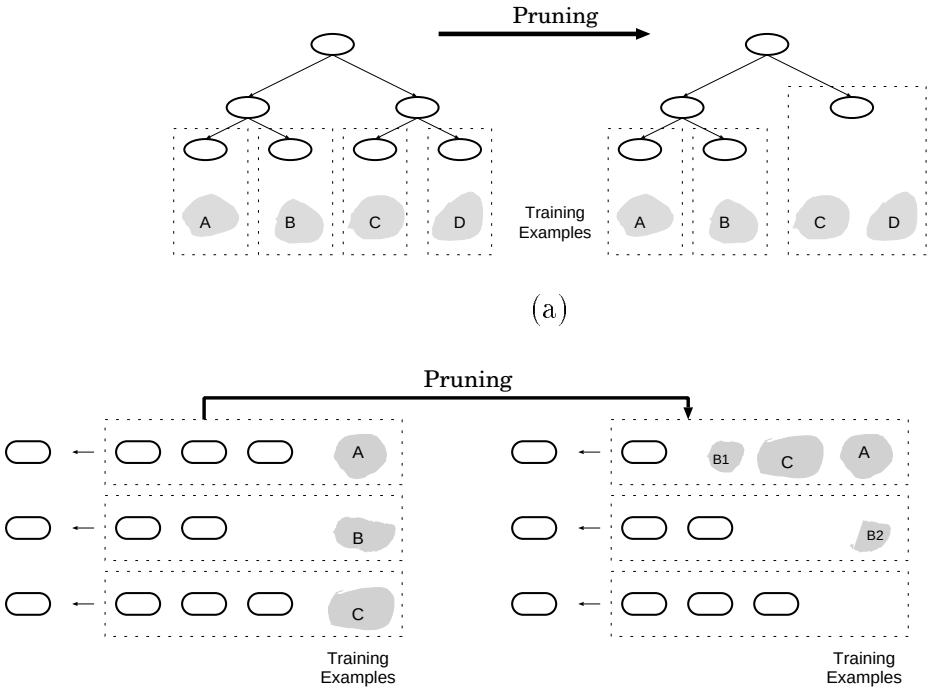

Figure 2: Post-Pruning in (a) divide-and-conquer and (b) separate-and-conquer learning algorithms.

subsequent clauses. However, the initial growing phase of REP does not know which of the instances are noisy and will henceforth carry along instances that should already be covered by one of the previous clauses.

In the example of figure 2 (b) the first of three rules is simplified and now covers not only the examples its original version has covered, but also all of the examples that the third rule has covered and some of the examples that the second rule has covered. While the third rule could easily be removed by the pruning algorithm, in general it need not be the case that the second rule or one of its pruned versions will be good explanations for the remaining set of examples B2, because B2 is a subset of the original set B and pruning operators can only generalize and not specialize the concept. Besides, it might well be the case that a good explanation for B2 consists of a totally different set of literals than a good explanation for its superset B. Conversely, if several sets like B2 appear, their union might have a much simpler explanation than the logical disjunction of the explanations previously found for their supersets.

Another way of looking at this problem may be to view a PROLOG program as a binary decision tree that allows conjunctive tests at each interior node, and

where at least one of the two successors of each node is a leaf. The body of each clause of the program corresponds to a node in the decision tree. If the body is true, the head is proven and we arrive at a leaf node. Otherwise we try the next node in the tree, i.e. the next clause in the program. Classical decision tree pruning would only allow pruning the nodes bottom up, i.e. only allow the deletion of clauses from the end of the program. REP, however, not only allows pruning any (instead of only the last) node, but also pruning the conditions of the rules associated with each node by deleting literals. Changing the test associated with a node in a decision tree will in general change the split it induces on the examples and thus could lead to the generation of different subtrees for its children. However, as the test is changed at pruning time (*after* learning), REP has to keep the subtree that has been previously learned from a different set of examples, although there might be a better subtree to explain this new set of examples.

Thus it is clear that the initial overfitting phase of post-pruning algorithms may in the best case only lead to the generation of some additional clauses that will be pruned in the pruning phase (like the third rule in the example). In the worst case, however, the instances that will be covered by a pruned rule, but are not covered by its unpruned original (the sets C and B1 in our example) may lead the learner down a garden path. They may change the evaluation of the candidate literals in subsequent learning and thus the "correct" literals might not be selected. A wrong choice of a literal cannot be undone by pruning.

3 I-REP

Incremental Reduced Error Pruning (I-REP) (Fürnkranz and Widmer 1994) was motivated by the observation that REP is incompatible with the separate-and-conquer learning strategy as we have discussed in section 2. Its basic idea is that instead of first growing a complete concept description and pruning it thereafter, each individual clause will be pruned right after it has been generated. This ensures that the algorithm can remove the training examples that are covered by the pruned clause before subsequent clauses are learned. Thus it can be avoided that these examples influence the learning of the following clauses.

A pseudo-code version of the algorithm can be found in Fig. 3. Before learning a clause, the current set of training examples is split into a growing (usually 2/3) and a pruning set (usually 1/3) as in many post-pruning algorithms. After learning a clause from the growing set, literals will be deleted from this clause in a greedy fashion until any further deletion would decrease the accuracy of this clause on the pruning set. The resulting rule will then be added to the concept description and all covered positive and negative examples will be removed from the training — growing *and* pruning — set. The remaining training instances are then redistributed into a new growing and a new pruning set to ensure that each

```
procedure I-REP(Examples, SplitRatio)

Theory = ∅
while POSITIVE(Examples) ≠ ∅
      Clause = ∅
      SPLITEXAMPLES(SplitRatio, Examples, GrowingSet, PruningSet)
      Cover = GrowingSet
      while NEGATIVE(Cover) ≠ ∅
            Clause = Clause ∪ FINDLITERAL(Clause, Cover)
            Cover = COVER(Clause, Cover)
      loop
           NewClause = SIMPLIFYCLAUSE(Clause, PruningSet)
           if ACCURACY(NewClause, PruningSet) <
                          ACCURACY(Clause, PruningSet)
              exit loop
           Clause = NewClause
      if ACCURACY(Clause, PruningSet) ≤
                    ACCURACY(fail, PruningSet)
         exit while
      Examples = Examples − Cover
      Theory = Theory ∪ Clause
return(Theory)
```

Figure 3: Incremental Reduced Error Pruning

of the two sets contains the predefined percentage of the remaining examples. From these sets the next clause will be learned. When the predictive accuracy of the pruned clause is below the predictive accuracy of the empty clause (i.e. the clause with the body `fail`), the clause will not be added to the concept description and I-REP returns the learned clauses.

Most of the efficiency of the I-REP algorithm comes from the integration of pre-pruning and post-pruning by this definition of a stopping criterion based on the accuracy of the pruned clause on the pruning set. Thus I-REP does not need REP's `delete-clause` operator (Brunk and Pazzani 1991), because the clauses of the final theory are constructed directly and learning stops when no more useful clauses can be found. However, this may also cause problems: Whenever the pruned clause does not have an accuracy above the accuracy of the empty clause, no more clauses will be learned. If this accuracy is not estimated accurately, either because there are not enough remaining examples or because of a bad split, I-REP will be prone to over-generalization.

```
procedure I²-REP(Examples)

Theory = ∅
while POSITIVE(Examples) ≠ ∅
      Clause = ∅
      SPLITEXAMPLES(0.5, Examples, SetA, SetB)
      CoverA = SetA
      CoverB = SetB
      while (NEGATIVE(CoverA) ≠ ∅) ∧ (NEGATIVE(CoverB) ≠ ∅)
            ClauseA = Clause ∪ FINDLITERAL(Clause, CoverA)
            ClauseB = Clause ∪ FINDLITERAL(Clause, CoverB)
            if ACCURACY(ClauseA,Examples) >
                        ACCURACY(ClauseB,Examples)
               NewClause = ClauseA
            else
               NewClause = ClauseB
            if ACCURACY(NewClause,Examples) <
                        ACCURACY(Clause,Examples)
               exit while
            CoverA = COVER(NewClause, CoverA)
            CoverB = COVER(NewClause, CoverB)
            Clause = NewClause
      if ACCURACY(Clause,Examples) ≤
                  ACCURACY(fail,Examples)
         exit while
      Examples = Examples − (CoverA ∪ CoverB)
      Theory = Theory ∪ Clause
return(Theory)
```

Figure 4: I²-REP

4 I²-REP

I-REP still has to learn overfitting clauses which we tried to avoid with a new algorithm. Just as I-REP improves upon REP by pruning on the clause level instead of the theory level, we tried to improve I-REP by pruning on the literal level instead of the clause level. Figure 4 shows the resulting algorithm, I²-REP.

As in pre-pruning algorithms I²-REP tries to select only the right literals in the first place and to decide when to stop adding literals to the theory. However, it uses a typical post-pruning method (evaluation on a separate pruning set) to do so. For this purpose the set of training examples is split into two subsets of equal size. A literal that maximizes some heuristic function is found for each of the two sets. These two literals are then compared and the one that has a higher accuracy on the entire set of examples is chosen to extend the current clause. This is repeated until the clause covers no negative examples in one of the two

sets or until the chosen literal does not improve the accuracy of this clause. In that case the learned clause is compared to the clause with the body `fail` and if its accuracy is higher, it will be added to the theory and the next clause will be learned from the examples that are not yet covered. If the current clause cannot improve upon the empty clause, learning stops as in I-REP.

One of the problems with I-REP is that a bad split of the training examples into a growing and a pruning set can cause over-generalisation, because I-REP would either learn an incorrect clause from a bad growing set or evaluate a correct clause on a bad pruning set. In both cases the learned clause may appear worse than the empty clause and I-REP will stop. As we will see in section 5 this can lead to the learning of over-general domain theories, in particular in domains with only a limited amount of noise or domains with low example set sizes. I^2-REP having two literals to chose from, will hopefully be less likely to prematurely stop learning if one of them is a bad choice or a good choice that is badly evaluated. Besides, I^2-REP's procedure for selecting a literal is very similar to 2-fold cross-validation which has recently been shown to be a reliable procedure for comparing classifiers, in particular at low training set sizes (Weiss and Indurkhya 1994). Therefore we hope that I^2-REP will be able to improve upon I-REP in these cases.

5 Experiments

We have tested REP, I-REP, and I^2-REP on the relational KRK chess endgame domain and on several propositional domains from the UCI repository of Machine Learning databases. I-REP was tested with two different settings. The one that has been used in all previous experiments (see e.g. (Fürnkranz 1994a)) splits the training examples in the ratio 2/3 growing and 1/3 pruning examples. In the other setting I-REP splits the training examples into halves. The latter version has been included into the test series in order to have a better evaluation of the merits of I^2-REP, because this version of I-REP used exactly the same splits as I^2-REP, but only selected literals from the first of the two sets.

All algorithms were implemented in SICStus PROLOG and had major parts of their implementations in common. In particular they shared the same interface to the data and used the same procedures for splitting the training sets. Mode, type and symmetry information about the background relations was used to restrict the search space. Information gain was used as a search heuristic. All results are averaged from 10 different runs on different training sets. We report the average accuracy, its standard deviation σ_n, and its range (the difference between the maximum and minimum value encountered in the 10 runs). The range information was used for a simple significance test which can be used to quickly determine significant differences between the averages for small ($n < 20$) sample sizes (Mittenecker 1977). For $n = 10$ the value of $L = \frac{\mu_1 - \mu_2}{R_1 + R_2}$ has to be

> 0.152 for a significance level of 5% and > 0.210 for a significance level of 1%, where μ_i are averages and R_i are ranges.

In addition, we also report the run-times of all algorithms. The results of REP are taken from previous experiments (Fürnkranz 1994a) which were performed on the same datasets, but on a slower machine. The values for the run-times of REP have therefore been adjusted by a factor that has been estimated from the run-time differences of I-REP on both machines. This has yielded very good approximations as we have confirmed with a few test runs.

5.1 Results in the KRK domain

The KRK domain has become a standard benchmark problem for testing relational learning systems. The domain theory used in our experiments is extensively described in the appendix of (Fürnkranz 1994a). In the KRK domain we used 5 different training set sizes containing 10% of artificial class noise, i.e. 10% of the training examples were misclassified. For each training set size we used 10 different example sets. Accuracies were measured on 5000 noise-free examples. The training and test data were the same for all algorithms. The data sets used are exactly the same as in our previous experiments in this domain (Fürnkranz 1994a; Fürnkranz 1994) so that the accuracy results are directly comparable to the results of other algorithms reported there. However, as already mentioned, the run-times have to be adjusted accordingly.

From table 1 we can see that I-REP and I^2-REP in general learn more accurate theories than REP. In fact, I^2-REP learns a significantly (1%) better theory than REP at all training set sizes. I-REP is not as consistent. In particular at low training set sizes it is likely to over-generalize. E.g. at size 100 I-REP (2/3) returns a theory that is significantly worse than the one learned by I^2-REP. At low training set sizes I^2-REP seems to be able to make better use of the available training examples. At high example set sizes I-REP and I^2-REP both learn simple approximations to the target concept, while REP starts to overfit the noise, because it has to prune more and more clauses and literals and is likely to get stuck at a local optimum.

I-REP (1/2) learns significantly better theories than REP at sizes 500 and 1000, while I-REP (2/3) does so at sizes 250 and 1000. In addition, at size 1000 I-REP (2/3) was significantly better than its counterpart that used only 1/2 of the examples for learning. However, in general the good performance of the latter version showed that it is not only important to have a enough examples for learning, but there also have to be enough examples for pruning. I^2-REP's way of using both example sets for both tasks seems to form a reasonable compromise.

However, there clearly is a price to be paid for this increase in reliability: I^2-REP is slower than I-REP. Nevertheless the asymptotic time complexity of both algorithms is about the same as can be seen from table 2 which contains a log-log analysis as originally suggested in (Cameron-Jones 1994). Dividing the

Table 1: Results in the KRK domain.

KRK-100	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**91.77**	8.59	28.52	5.68
I-REP (2/3)	**84.55**	9.39	31.00	2.21
I-REP (1/2)	**91.17**	9.78	31.44	2.04
I^2-REP	**95.48**	5.12	14.32	3.14
KRK-250	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**96.29**	1.84	6.30	101.10
I-REP (2/3)	**98.34**	0.89	3.66	8.91
I-REP (1/2)	**96.68**	4.12	13.78	6.25
I^2-REP	**98.48**	0.66	2.30	11.60
KRK-500	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**97.62**	0.91	3.12	1025.71
I-REP (2/3)	**98.49**	1.24	4.56	23.35
I-REP (1/2)	**98.80**	0.54	1.36	18.19
I^2-REP	**98.92**	0.59	1.50	27.83
KRK-750	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**97.47**	1.36	4.62	4131.84
I-REP (2/3)	**98.86**	0.48	1.22	40.89
I-REP (1/2)	**99.19**	0.42	1.16	35.19
I^2-REP	**99.21**	0.42	1.28	49.89
KRK-1000	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**97.72**	1.18	3.70	13615.50
I-REP (2/3)	**99.48**	0.25	0.92	59.10
I-REP (1/2)	**98.94**	0.55	1.36	44.80
I^2-REP	**99.27**	0.50	1.56	71.10

differences of the logarithms of the run-times by the differences of the logarithms of the training set sizes gives an approximation for the degree of the highest order term of the growth function. We have tabulated the slopes for adjacent training set sizes in table 2.

The main result for our study is that I-REP and I^2-REP both have a sub-quadratic time complexity in noisy domains. This is consistent with the conjecture of (Fürnkranz and Widmer 1994), where we estimated I-REP to have a time complexity of $O(n \log^2 n)$ on random data. Thus both algorithms are significantly faster than the initial overfitting phase of post-pruning algorithms which typically has a time complexity of $\Omega(n^2 \log n)$ (Cohen 1993) (see also the first column of table 2). Asymptotically I^2-REP even seems to be a little more efficient than I-REP, as can be expected from the fact that it does not have to learn overfitting clauses. However, the presented results do not allow this conclusion.

Our results also confirm that the pruning phase is the most expensive part of REP with a time complexity of $\Omega(n^4)$. However, in (Cohen 1993), where this problem was discussed for the first time, an efficient alternative was suggested

Table 2: Log-log analysis of the run-times on noisy KRK data.

KRK Set Sizes	REP Rule Growth	REP Pruning	I-REP (2/3)	I-REP (1/2)	I^2-REP
100-250	2.61	4.11	1.52	1.22	1.20
250-500	2.32	3.91	1.39	1.54	1.26
500-750	2.26	3.78	1.38	1.63	1.44
750-1000	2.16	4.00	1.28	0.84	1.23

that keeps the run-time of post-pruning below the run-time of the initial rule growing phase by using a top-down instead of a bottom-up search.[1] Although this would speed up the total learning time considerably, the algorithm would still be much slower than I-REP and I^2-REP, because their run-times are significantly below that of the initial overfitting phase that is common to all post-pruning approaches.

5.2 Propositional Domains

We have also compared the algorithms on various propositional data sets from the UCI repository of Machine Learning databases. In this case our algorithms behave very similar to the CN2 rule induction system (Clark and Niblett 1989; Clark and Boswell 1991). The appendix of (Holte 1993) gives a summary of the results achieved by various algorithms on some of the most commonly used data sets of the UCI repository and a short description of these sets. We selected 9 of them for our experiments. The remaining sets were not used because either the description of the data sets was unclear or they had more than two classes, which cannot be handled by our current implementation of the learning algorithms. In the *Lymphography* data set we removed the 6 examples for the classes "normal find" and "fibrosis" in order to get a 2-class problem. All other data were used as described in (Holte 1993).

In all datasets the background knowledge consisted of < and = relations with one variable and one constant argument. Wherever appropriate, comparisons between two different variables of the same data type were allowed as well (e.g. in the *Vote* domains). Introduction of new variables was not allowed. For all

[1] Similar experiments reported in (Fürnkranz 1994a) confirm the result of (Cameron-Jones 1994) that the asymptotic time complexity of this top-down approach to pruning is still above the complexity of the initial overfitting phase contrary to a claim in (Cohen 1993). However, in absolute terms the pruning costs of the top-down version were always neglible compared to the costs of initial rule growing, while the opposite was true for the bottom-up pruning used in REP.

Table 3: Results in some propositional domains.

Breast Cancer	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**69.97**	3.80	12.16	104.35
I-REP (2/3)	**70.89**	5.23	19.58	11.75
I-REP (1/2)	**69.42**	4.51	14.48	6.51
I^2-REP	**70.93**	3.71	11.54	16.18
Hepatitis	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**76.96**	3.93	10.80	40.88
I-REP (2/3)	**78.66**	2.80	7.34	24.14
I-REP (1/2)	**78.09**	2.48	7.55	15.13
I^2-REP	**74.70**	6.27	20.11	49.21
Sick Euthyroid	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**97.55**	0.32	1.06	5121.28
I-REP (2/3)	**97.48**	0.50	1.70	986.31
I-REP (1/2)	**97.56**	0.41	1.45	507.11
I^2-REP	**97.61**	0.36	1.07	1367.22
Glass (G2)	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**77.76**	4.31	14.73	93.31
I-REP (2/3)	**76.31**	4.89	15.95	26.54
I-REP (1/2)	**74.06**	4.51	13.33	16.72
I^2-REP	**77.98**	4.77	14.62	36.24
Votes	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**95.84**	1.39	3.92	22.83
I-REP (2/3)	**94.75**	1.75	6.95	8.92
I-REP (1/2)	**95.27**	1.03	3.77	6.55
I^2-REP	**94.98**	1.14	3.88	9.84
Votes (VI)	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**86.72**	3.46	10.78	62.31
I-REP (2/3)	**87.25**	3.27	10.75	14.80
I-REP (1/2)	**86.51**	3.68	12.77	10.65
I^2-REP	**88.60**	2.06	8.07	22.63
KRKPa7	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**97.84**	0.54	2.01	3946.39
I-REP (2/3)	**97.74**	0.36	1.32	1660.65
I-REP (1/2)	**97.37**	0.57	1.75	1266.62
I^2-REP	**97.83**	0.49	1.86	2230.70
Lymphography (2 classes)	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**81.85**	4.86	16.83	7.72
I-REP (2/3)	**79.17**	4.42	15.30	4.16
I-REP (1/2)	**78.93**	5.24	16.15	3.18
I^2-REP	**82.32**	4.70	17.41	6.90
Mushroom	Accuracy	Stnd. Dev.	Range	CPU secs.
REP	**99.97**	0.05	0.15	763.82
I-REP (2/3)	**99.97**	0.04	0.11	986.04
I-REP (1/2)	**99.96**	0.05	0.15	856.73
I^2-REP	**99.94**	0.07	0.19	1373.26

data sets the task was to learn a definition for the minority class. All experiments followed the setup used in (Holte 1993), i.e. the algorithms were trained on 2/3 of the data and tested on the remaining 1/3. However, only 10 runs were performed for each algorithm on each data set.

In the results of table 3 I^2-REP achieves the highest accuracy in 5 of the 9 data sets. However, none of the differences in accuracy are significant, not even at the 5% level (mostly because the results of different runs in the same domain vary considerably). In the *Hepatitis* domain I^2-REP is clearly worse than the other algorithms. In this domain I-REP profits from its strong overfitting avoidance bias. In some of the 10 runs it learns an empty theory and classifies it with baseline accuracy. In this domain this helps to improve the average performance, because all algorithms learn theories that are below the default accuracy. That I^2-REP's overfitting avoidance bias is not as strong as I-REP's can also be seen from the results in domains with low noise levels[2] like *KRKPa7* and *Lymphography*, where I^2-REP achieves better results. In these domains, as in domains with low example set sizes, I-REP is likely to over-generalize. This can also be seen from the run-times when I^2-REP is more than twice as slow as I-REP (1/2), which means that it must have learned more literals.

I-REP is the fastest algorithm in all tested domains except for the noise-free *Mushroom* domain. Here algorithms that do not prune at all are able to achieve an accuracy of 100% (see (Fürnkranz 1994)). REP's post-pruning phase therefore does not change much on the theories produced by the initial overfitting phase and therefore is almost costless. The overhead associated with I-REP and I^2-REP in this case outweighs the benefits (I^2-REP also seems to overfit the data a little bit). But in general I-REP has proven to be a more efficient alternative to REP in noisy domains, while I^2-REP improves upon I-REP in domains with low noise levels and/or low example set sizes.

6 Conclusion

In this paper we have outlined some principal problems with using *Reduced Error Pruning* for separate-and-conquer rule learning algorithms. We have suggested two alternative versions that, by integrating pruning and learning, do not prune entire domain theories, but prune at the clause level (I-REP) or even at the literal level (I^2-REP). Doing so results in considerable gains in efficiency (both alternatives are significantly faster than REP's initial rule growing phase alone) and may also yield better results in terms of accuracy by avoiding the above-

[2] In (Fürnkranz 1994a) and (Fürnkranz 1994) we grouped the nine domains into three sets: In the first three domains post-pruning significantly improved upon the result from the initial overfitting phase, in the second three domains it did not change much and in the last three domains it worsens the result. We have interpreted this as evidence that the last three domains are not very noisy, while the first three are.

mentioned problems. However, the differences in the tested propositional domains were not as significant as in the relational KRK domains, possibly because the discussed problems are not as relevant for the former as they are for the latter.

Recently several alternatives to REP have been proposed (Cohen 1993; Cameron-Jones 1994). In (Fürnkranz 1994a) we discuss some of these algorithms in detail and compare them to I-REP as well as to other relational or propositional learning systems. However, as they are mostly concerned with improving REP's expensive pruning phase only, they are also subject to the problems discussed in section 2.

Acknowledgements

This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P10489-MAT. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Research. I would like to thank Gerhard Widmer and Bernhard Pfahringer for many helpful comments and discussions. Thanks are also due to William Cohen, Mike Cameron-Jones and Ross Quinlan for some valuable suggestions.

References

Brunk, C. A. and M. J. Pazzani (1991). An investigation of noise-tolerant relational concept learning algorithms. In *Proceedings of the 8th International Workshop on Machine Learning*, Evanston, Illinois, pp. 389–393.

Cameron-Jones, R. (1994, May). The complexity of Cohen's grow method. Unpublished draft for comments.

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

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

Cohen, W. W. (1993). Efficient pruning methods for separate-and-conquer rule learning systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, pp. 988–994.

Džeroski, S. and I. Bratko (1992). Handling noise in Inductive Logic Programming. In *Proceedings of the International Workshop on Inductive Logic Programming*, Tokyo, Japan.

Fürnkranz, J. (1994). A comparison of pruning methods for relational concept learning. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pp. 371–382.

Fürnkranz, J. (1994a). *Efficient Pruning Methods for Relational Learning.* Ph. D. thesis, Vienna University of Technology.

Fürnkranz, J. (1994b). FOSSIL: A robust relational learner. In *Proceedings of the European Conference on Machine Learning*, Catania, Italy, pp. 122–137. Springer-Verlag.

Fürnkranz, J. (1994). Pruning methods for rule learning algorithms. In *Proceedings of the 4th International Workshop on Inductive Logic Programming*, Number 237 in GMD–Studien, pp. 321–336.

Fürnkranz, J. and G. Widmer (1994). Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, pp. 70–77.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning 11*, 63–91.

Mittenecker, E. (1977). *Planung und statistische Auswertung von Experimenten* (8th ed.). Vienna, Austria: Verlag Franz Deuticke. In German.

Pagallo, G. and D. Haussler (1990). Boolean feature discovery in empirical learning. *Machine Learning 5*, 71–99.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning 1*, 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies 27*, 221–234.

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

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

Weiss, S. M. and N. Indurkhya (1994). Small sample decision tree pruning. In *Proceedings of the 11th Conference on Machine Learning*, Rutgers University, New Brunswick, NJ, pp. 335–342.