Pruning Methods for Rule Learning Algorithms

Johannes Fürnkranz
Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna
Austria
E-mail: `juffi@ai.univie.ac.at`

Abstract

In this paper we will shortly review several pruning methods for relational learning algorithms and show how they are related to each other. We then report some experiments in several natural domains and try to analyse the performance of the algorithms in these domains in terms of run-time and accuracy. While some algorithms are clearly faster than others, no safe recommendation for achieving high accuracy can be given.

1 Introduction

Lately several pruning methods for noise handling in relational rule learning algorithms have been proposed. The classic approaches to pruning are based on *pre-pruning* (Foil [Quinlan, 1990], mFoil [Džeroski and Bratko, 1992], or Fossil [Fürnkranz, 1994b]) and *post-pruning* (*Reduced Error Pruning* (REP) [Brunk and Pazzani, 1991] and Grow [Cohen, 1993]). More recently approaches have been proposed that *combine* (MDL-Grow [Cohen, 1993] and *Top Down Pruning* (TDP) [Fürnkranz, 1994c]) and *integrate* (*Incremental Reduced Error Pruning* (I-REP) [Fürnkranz and Widmer, 1994]) these two basic methods.

We will present and discuss a variety of these pruning algorithms in section 2 and in particular show how they are related to each other. In section 3 we report experiments with these algorithms that we have performed in a variety of natural domains. The results, in particular those concerning the accuracy of the various algorithms will turn out to be very diverse. We will nevertheless draw some conclusions in section 4.

2 Pruning in Relational Learning Algorithms

Figure 1 shows a typical separate-and-conquer rule learning algorithm. Literals are added to the body of a clause until no more negative examples are covered. Clauses are added to the program until all of the positive examples are covered.

```
procedure SeparateAndConquer(Examples)

Program = $\emptyset$
while Positive(Examples) $\neq \emptyset$
    Clause = $\emptyset$
    Cover = Examples
    while Negative(Cover) $\neq \emptyset$
        Clause = Clause $\cup$ FindLiteral(Clause, Cover)
        Cover = Cover(Clause, Cover)
    Examples = Examples $-$ Cover
    Program = Program $\cup$ Clause
return(Program)
```

Figure 1: A Separate-and-Conquer rule learning algorithm

This algorithm works fine as long as the given background knowledge is guaranteed to be correct. However, in most natural domains the data may contain misclassifications or errors in the attributes. In these cases it is not good to learn a concept description that perfectly explains all of the positive and none of the negative examples, because the algorithm would try to find explanations for the erroneous examples as well. This problem is known as *overfitting the noise*.

Pruning is a standard way of dealing with the problem of overfitting (see e.g. [Mingers, 1989, Esposito *et al.*, 1993]). There are two fundamentally different approaches [Cestnik *et al.*, 1987], *Pre-Pruning* and *Post-Pruning*. In sections 2.1 and 2.2 we will review classical pruning methods that have been adopted for relational concept learning systems. In section 2.3 we show how post-pruning can be combined with a pre-pruning phase in order to get starting theory with a reduced amount of overfitting. Finally, in section 2.4, we introduce a method that integrates both approaches by using post-pruning methods as a pre-pruning criterion.

2.1 Pre-Pruning

Pre-Pruning methods deal with noise during concept generation. Algorithms that have no pre-pruning heuristics are typically subject to overfitting the noise in the data. They try to find concept descriptions that perfectly explain all the positive examples, but do not cover any negative examples. In noisy data sets

```
procedure PREPRUNING(Examples)

Program = ∅
while POSITIVE(Examples) ≠ ∅
        Clause = ∅
        Cover = Examples
        while NEGATIVE(Cover) ≠ ∅
                NewClause = Clause ∪ FINDLITERAL(Clause, Cover)
                if STOPPINGCRITERION(Program,NewClause,Cover)
                   exit while
                Clause = NewClause
                Cover = COVER(Clause,Cover)
        Examples = Examples − Cover
        Program = Program ∪ Clause
return(Program)
```

Figure 2: A Pre-Pruning algorithm

they therefore learn overly specific clauses which often explain only a few noisy examples and thus are bad generalizations. By learning a few "over-general" rules instead of many very specific clauses, these algorithms deliberately cover some negative training examples and leave some positive training examples uncovered in order to learn rules with a higher predictivity.

Figure 2 shows a typical modification to basic separate-and-conquer rule learning algorithm that includes a so-called *stopping criterion*, i.e. a heuristic that determines when to stop adding conditions to a rule, and when to stop adding rules to the concept description. The only difference to the basic algorithm of figure 1 is that if the current clause fulfills the stopping criterion the inner `while` loop is terminated and the incomplete clause will be added to the concept description. Note that if this clause is empty, no positive (and no negative) examples are covered and the outer loop will terminate as well.

The most commonly used stopping criteria are

- *Encoding Length Restriction*: This heuristic used in the classic ILP system FOIL [Quinlan, 1990] is based on the Minimum Description Length principle [Rissanen, 1978]. It prevents overfitting the noise by learning only as long as the costs of encoding the current clause are less than the costs of encoding the tuples covered by it.

- *Significance Testing* was first used in [Clark and Niblett, 1989] and later on in the ILP system mFOIL [Džeroski and Bratko, 1992]. It tests for significant differences between the distribution of positive and negative examples covered by a rule and the overall distribution of positive and negative examples by comparing the likelihood ratio statistic to a χ^2 distribution with

1 degree of freedom at the desired significance level. Insignificant rules are rejected.

- *Cutoff Stopping Criterion*: This simple method used in FOSSIL [Fürnkranz, 1994b] only allows to add conditions to a rule when their heuristic values are above a predefined threshold. Thus this *cutoff parameter* allows the user to directly control the amount of overfitting. At $Cutoff = 0.0$ the algorithm will fit all of the data (no pre-pruning), while at $Cutoff = 1.0$ FOSSIL will fit none of the data and learn the empty theory (maximum pre-pruning). A value of $Cutoff \approx 0.3$ has proved to be very robust with respect to varying noise levels and training set sizes [Fürnkranz, 1994b].

mFOIL's significance testing along with the m-estimate and a powerful beam search have been very successful in learning concepts in noisy domains [Džeroski and Bratko, 1992]. Similar results have been obtained for the very efficient cutoff criterion. Both have been shown to be superior to the encoding length restriction, because the latter is dependent on the size of the training set, so that the size of the learned concepts (and thus the amount of overfitting) may increase with training set size [Fürnkranz, 1994b].

2.2 Post-Pruning

Post-pruning was introduced to relational learning algorithms with *Reduced Error Pruning* (REP) [Brunk and Pazzani, 1991] based on previous work by [Quinlan, 1987] and [Pagallo and Haussler, 1990]. The basic idea is that in a first pass, no attention is paid to the noise in the data and a concept description that explains all of the positive and none of the negative examples is learned. For this purpose the training set is split into two subsets: a *growing set* (usually 2/3) and a *pruning set* (1/3). The concept description that has been learned from the growing set is then simplified by greedily deleting conditions and rules from the theory until any further deletion would result in a decrease of predictive accuracy measured on the pruning set.

However, this approach has several disadvantages, most notably efficiency. [Cohen, 1993] has shown that REP has a time complexity of $\Omega(n^4)$ on purely random data. Therefore [Cohen, 1993] proposed GROW, a new pruning algorithm based on a technique used in the GROVE learning system [Pagallo and Haussler, 1990]. Like REP, GROW first finds a theory that overfits the data. But instead of pruning the intermediate theory until any further deletion results in a decrease of accuracy on the pruning set, generalizations of clauses from this theory are subsequently selected to form the final concept description until no further clause will improve predictive accuracy on the pruning set. Thus GROW performs a general-to-specific search instead of REP's specific-to-general search. For noisy data the asymptotic costs of this pruning algorithm have been shown to be below the costs of the initial phase of overfitting.

```
procedure PostPruning(Examples, SplitRatio)

SplitExamples(SplitRatio, Examples, GrowingSet, PruningSet)
Program = SeparateAndConquer(GrowingSet)
loop
    NewProgram = SimplifyProgram(Program,PruningSet)
    if Accuracy(NewProgram,PruningSet) < Accuracy(Program,PruningSet)
      exit loop
    Program = NewProgram
return(Program)
```

Figure 3: A Post-Pruning algorithm

2.3 Combining Pre- and Post-Pruning

As stated in the last section, the Grow algorithm can drastically reduce the costs of pruning an overly specific theory. However, the overall costs of the algorithm are still unnecessarily high, because like REP, Grow has to learn an overly specific intermediate theory. A logical improvement would therefore be to limit the amount of overfitting by replacing the call to SeparateAndConquer with a call to PrePruning in the algorithm of figure 3. [Cohen, 1993] does this with the Grow algorithm using two weak MDL-based stopping criteria. These methods are not intended to entirely prevent overfitting like the pre-pruning approaches of section 2.1, but to reduce the amount of overfitting, so that the post-pruning phase can start off with a better theory and has to do less work.

However, there is always the danger that a predefined stopping criterion will over-generalize the theory. To avoid this [Fürnkranz, 1994c] have developed an algorithm called *Top-Down Pruning* (TDP) (see figure 4). This method searches the theories that can be generated with different settings of Fossil's cutoff parameter (see section 2.1) in a top-down general-to-specific fashion and selects the most specific theory within one standard error of classification of the most accurate theory as a starting point for the post-pruning phase.[1] The hope is that this theory will not be an over-generalisation (it is more specific than the most accurate theory found so far), but will also be close to the intended theory (its accuracy is still close to the best so far), so that only a limited amount of pruning has to be performed. The implementation of TDP made use of several optimizations, so that finding this theory is often cheaper than fitting the noise. A more detailed description of this process can be found in [Fürnkranz, 1994c].

[1]This method is inspired by the approach taken in CART [Breiman *et al.*, 1984] where the most general decision tree within this standard error margin is selected as a final theory.

```
procedure TDP(Examples, SplitRatio)

Cutoff = 1.0
BestProgram = ∅
BestAccuracy = 0.0
SplitExamples(SplitRatio, Examples, GrowingSet, PruningSet)
repeat
        NewProgram = Fossil(GrowingSet, Cutoff)
        NewAccuracy = Accuracy(NewProgram, PruningSet)
        if NewAccuracy > BestAccuracy
           BestProgram = NewProgram
           BestAccuracy = NewAccuracy
           LowerBound = BestAccuracy − StandardError(BestAccuracy, PruningSet)
        Cutoff = MaximumPrunedCorrelation(NewProgram)
until (NewAccuracy < LowerBound) or (Cutoff = 0.0)
loop
      NewProgram = SimplifyProgram(Program, PruningSet)
      if Accuracy(NewProgram, PruningSet) < Accuracy(Program, PruningSet)
         exit loop
      Program = NewProgram
return(Program)
```

Figure 4: Combining Pre- and Post-Pruning with *Top-Down Pruning*.

2.4 Integrating Pre-and Post-Pruning

There are several problems with pruning in relational concept learning (see [Fürnkranz and Widmer, 1994]). Not all of them are attacked by the algorithms in the previous sections. In particular, the separate-and-conquer strategy (see figure 1) may cause problems. The important difference between this method and the divide-and-conquer strategy used in most decision tree learning algorithms is that pruning of branches in a decision tree will never affect the neighboring branches, whereas pruning of conditions of a rule will affect all subsequent rules. Deleting a condition from a rule means that the clause is generalized, i.e. it will cover more positive instances along with some negative instances. Consequently these additional instances should be removed from the training set so that they cannot influence the learning of subsequent clauses. However, the initial growing phase of post-pruning algorithms does not know which of the instances will be covered by the pruned rule and is therefore not able to remove them from the training set. In the best case those superfluous examples in the growing phase only lead to the generation of some additional clauses that will be pruned in the pruning phase. In the worst case, however, those instances may lead the learner down a garden path, because they may change the evaluation of the candidate relations in subsequent learning and thus the "correct" literals might not

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

Program = $\emptyset$
while Positive(Examples) $\neq \emptyset$
        Clause = $\emptyset$
        SplitExamples(SplitRatio, Examples, GrowingSet, PruningSet)
        Cover = GrowingSet
        while Negative(Cover) $\neq \emptyset$
                Clause = Clause $\cup$ 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) $\leq$ Accuracy(fail, PruningSet)
           exit while
        Examples = Examples $-$ Cover
        Program = Program $\cup$ Clause
return(Program)
```

Figure 5: Integrating Pre- and Post Pruning with *Incremental Reduced Error Pruning*

be selected. A wrong choice of a literal cannot be undone by pruning.

Another way of viewing this problem might be to imagine each clause of the PROLOG program as a node in a decision list, i.e. a binary decision tree, where each node's children contain at least one leaf. Classical decision tree pruning would only allow to prune the nodes bottom up, i.e. only delete the last rule. The version of REP described in section 2.2 not only allows to prune any (instead of only the last) node, but also to prune the conditions of the rules associated with each node.

Incremental Reduced Error Pruning (I-REP) [Fürnkranz and Widmer, 1994] is an efficient solution to this problem by means of integrating pre-pruning and post-pruning: Each clause is learned until it covers no more negative examples. Then literals are deleted from this clause in a greedy fashion until any further deletion would decrease the accuracy of this clause on a separate pruning set. The resulting rule is then added to the concept description and all covered positive and negative examples are removed from the training — growing *and* pruning — set. The remaining instances in the training set are then redistributed into a growing and a pruning set and a new clause is 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 is not added to the concept description and I-REP returns the learned clauses. Thus the accuracy of a clause on the pruning set also serves as a stopping criterion, i.e. post-pruning methods are used as a pre-pruning heuristic.

It can be easily seen that I-REP integrated pre-pruning (figure 2) and post-pruning (figure 3) into a single algorithm.

3 Experiments

In previous research [Fürnkranz, 1994a] we have compared a variety of these algorithms in the well-known noisy KRK endgame classification task [Muggleton *et al.*, 1989] at several different training set sizes. The experiments showed that the post-pruning algorithms are very inefficient, while pre-pruning is fast, but less accurate. I-REP, the integration of pre- and post-pruning, seems to unite the advantages of both methods. It was the most accurate algorithm and was only a little slower than the pre-pruning approaches.

Figure 6 (taken from [Fürnkranz, 1994a]) shows the accuracy (with standard deviations) of learning from 1000 training examples (10% of them misclassified) plotted against the logarithm of the run-time. The accuracy of the learned concepts was estimated on noise-free example sets of size 5000.

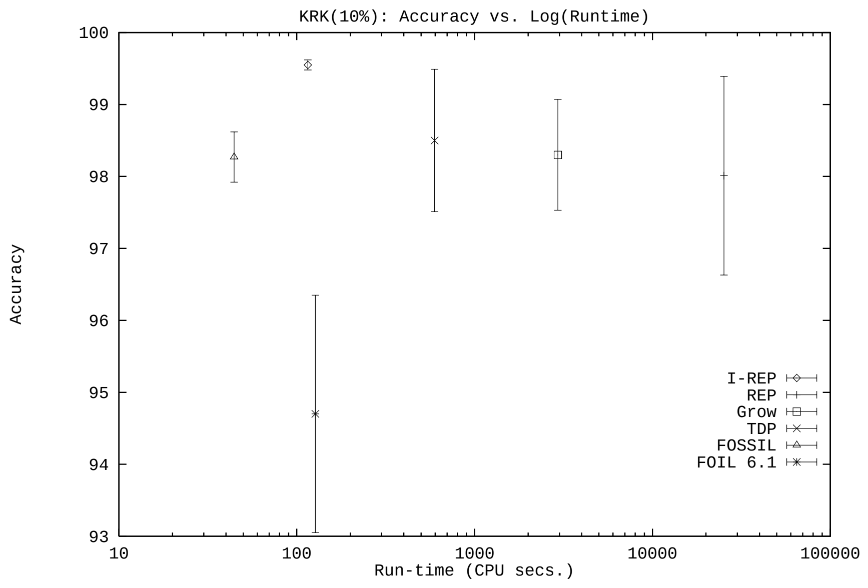

Figure 6: A comparison in the relational KRK domain (10% noise).

Although these results seem to be confirmed by similar experiments in the natural domain of finite element mesh design [Fürnkranz and Widmer, 1994, Fürnkranz, 1994c], we wanted to compare these algorithms on a variety of natural data sets in order to get more information about their applicability to real-world problems. Unfortunately we are not aware of any real-world relational test problems that are publicly available (except for the finite element mesh design data mentioned above). Therefore we had to use data from the UCI repository of Machine Learning databases that have been used to compare propositional learning algorithms. Propositional data do not challenge the underlying relational learning algorithms to their full extent. However, we are primarily concerned with the comparison of pruning methods, and for these it should not make much difference what sort of conditions are pruned, in particular for those algorithms that clearly separate the learning and pruning phases. Some of the issues discussed here might also be relevant for decision list learning algorithms such as those suggested in [Pagallo and Haussler, 1990].

A side effect of using propositional data is that we can compare propositional and relational learning algorithms and confirm that the quality of the concepts learned by the latter are not below those learned by the former (see [Cameron-Jones and Quinlan, 1993] for more experiments along these lines). 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 *Lymphograpy* 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]. For all data sets the task was to learn a definition for the minority class.

All algorithms tested in this study were implemented by the author 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 whenever possible. 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. Introduction of new variables was not allowed. In all experiments the value of FOSSIL's cutoff parameter has been set to 0.3. Run-times for all datasets were measured in CPU seconds for SUN SPARCstations ELC except for the *Mushroom* and *KRKPa7* datasets which are quite big and thus had to be run on a considerably faster SPARCstation S10. 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.

The results can be found in tables 1, 2, and 3. Each line shows the average accuracy on the 10 sets, its standard deviation and range (difference between the maximum and the minimum accuracy encountered), and the run-time of the algorithm. The results of C4.5 [Quinlan, 1993] are taken from the experiments

Breast Cancer	Accuracy	Stnd. Dev.	Range	Time
C4.5	71.96	4.36	—	—
Fossil	73.33	4.56	17.66	19.68
No Pruning	65.39	4.21	13.27	169.70
REP	69.97	3.80	12.16	257.29
Grow	68.46	4.72	15.39	183.67
No Pruning (TDP)	67.98	5.56	20.62	154.05
TDP	71.74	3.79	12.43	173.31
I-REP	70.89	5.23	19.58	28.97
Hepatitis	Accuracy	Stnd. Dev.	Range	Time
C4.5	81.23	5.12	—	—
Fossil	76.07	5.77	23.43	217.40
No Pruning	73.66	4.99	17.12	101.66
REP	76.96	3.93	10.80	102.28
Grow	76.45	4.24	11.14	102.39
No Pruning (TDP)	76.33	3.40	10.92	115.41
TDP	79.42	3.88	11.87	116.24
I-REP	78.66	2.80	7.34	60.40
Sick Euthyroid	Accuracy	Stnd. Dev.	Range	Time
C4.5	97.69	0.40	—	—
Fossil	97.58	0.40	1.35	891.40
No Pruning	96.25	0.51	1.70	4554.65
REP	97.55	0.32	1.06	5040.23
Grow	97.52	0.47	1.64	4635.26
No Pruning (TDP)	97.37	0.51	1.78	2965.51
TDP	97.49	0.43	1.21	3010.97
I-REP	97.48	0.50	1.70	970.70

Table 1: Results of the Breast Cancer, Hepatitis, and Sick Euthyroid domains.

Glass (G2)	Accuracy	Stnd. Dev.	Range	Time
C4.5	74.26	6.61	—	—
Fossil	77.32	4.79	15.96	216.42
No Pruning	75.24	5.26	18.15	91.89
REP	77.76	4.31	14.73	93.31
Grow	75.63	4.69	16.97	93.11
No Pruning (TDP)	77.23	4.01	12.64	85.56
TDP	75.90	6.18	20.51	87.39
I-REP	76.31	4.89	15.95	63.01
Votes	Accuracy	Stnd. Dev.	Range	Time
C4.5	95.57	1.31	—	—
Fossil	95.35	1.17	3.34	105.22
No Pruning	94.69	1.89	6.55	50.45
REP	95.84	1.39	3.92	57.41
Grow	95.63	1.36	3.92	53.84
No Pruning (TDP)	95.33	1.22	4.48	60.88
TDP	95.22	1.54	4.49	62.17
I-REP	94.75	1.75	6.95	22.43
Votes (VI)	Accuracy	Stnd. Dev.	Range	Time
C4.5	89.36	2.45	—	—
Fossil	89.07	2.64	8.13	88.94
No Pruning	86.46	2.01	7.36	124.47
REP	86.72	3.46	10.78	163.26
Grow	87.49	3.35	10.93	137.49
No Pruning (TDP)	87.57	1.36	4.29	105.67
TDP	85.85	2.62	9.21	113.05
I-REP	87.25	3.27	10.75	38.78

Table 2: Results of the Glass and Votes domains.

KRKPa7	Accuracy	Stnd. Dev.	Range	Time
C4.5	99.19	0.27	—	—
Fossil	95.17	2.66	8.63	2383.61
No Pruning	97.92	0.58	1.85	4063.80
REP	97.84	0.54	2.01	4243.08
Grow	97.48	0.41	1.06	4219.00
No Pruning (TDP)	96.26	1.85	4.74	2368.28
TDP	96.41	1.87	4.74	2376.28
I-REP	97.74	0.36	1.32	1785.50
Lymphography (2 classes)	Accuracy	Stnd. Dev.	Range	Time
C4.5 (on all 4 classes)	77.52	4.46	—	—
Fossil	87.22	4.39	17.23	20.79
No Pruning	83.25	4.79	16.03	17.05
REP	81.85	4.86	16.83	18.81
Grow	82.10	5.28	17.53	18.42
No Pruning (TDP)	83.73	5.50	17.53	18.66
TDP	81.86	4.39	12.39	20.27
I-REP	79.17	4.42	15.30	10.14
Mushroom	Accuracy	Stnd. Dev.	Range	Time
C4.5	100.00	0.00	—	—
Fossil	99.96	0.03	0.11	3538.19
No Pruning	100.00	0.01	0.04	1878.51
REP	99.97	0.05	0.15	1931.75
Grow	99.57	0.66	1.56	2088.81
No Pruning (TDP)	100.00	0.01	0.04	4595.23
TDP	99.97	0.05	0.15	4656.31
I-REP	99.97	0.04	0.11	2493.77

Table 3: Results of the Chess (KRKPa7), Lymphography, and Mushroom domains.

performed in [Holte, 1993] and are meant as an indicator of the performance of state-of-the-art decision tree learning algorithms on these data sets.

A short look shows that the results vary in terms of accuracy, but are quite consistent in run-times: I-REP is the fastest algorithm in 6 of the 9 test problems, while it is second-best in 2 of the remaining 3. The tables also confirm that GROW is usually faster than REP. TDP's results are not consistent, but it is faster than REP and GROW in some cases, which indicates that its initial top-down search for a good starting theory does not overfit the data as much as the initial rule growing phase of REP and GROW does. FOSSIL's run-times are very unstable. It is the fastest algorithm on some datasets, but by far the slowest on other data sets. The explanation for this probably lies in the fact that all algorithms except for FOSSIL only learn from 2/3 of the training data and use the remaining 1/3 for pruning. If not much pruning has to be done, it can be expected that FOSSIL is slower than the other algorithms.

Most differences in accuracies are not statistically significant.[2] Significant differences can be found in the *KRKPa7* chess endgame domain, where TDP and FOSSIL performed significantly (1%) worse than all other algorithms. FOSSIL was significantly (5%) better than TDP in the *Votes (VI)* domain[3] and outperformed (5%, sometimes 1%) all other algorithms in the *Lymphography* domain. In general C4.5 seems to be a little superior to the other algorithms (one cannot count the results on *Lymphography* where the rule learning algorithms had a presumably easier 2-class task.). However, the relational algorithms seem to be competitive.

To allow a more structured analysis we have grouped the 9 domains into 3 subclasses: Table 1 contains all domains where overfitting seems to be harmful, i.e. where REP's post-pruning phase significantly (at least 5%) improves upon the concepts learned by the initial overfitting phase.[4] Table 2 contains domains where pruning does not make a significant difference and finally table 3 contains all domains where pruning cannot be recommended as exemplified by the *Mushroom* data, where the overfitting phases learned 100% correct concept descriptions that were significantly better (5%) than those learned by all pruning algorithms. The *Mushroom* and *KRKPa7* domains are known to be not noisy, while the medical domains of table 1 are noisy. Therefore we suspect that our grouping of the domain somewhat corresponds to the amount of noise contained in the data.

[2]We have used a range test which can be used to quickly determine significant differences between medium values 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 of a significance level of 5% and > 0.210 for a significance level of 1%. (μ_i are medium values and R_i are ranges. Both can be found in tables 1 – 3.)

[3]This is the *Votes* domain with the most significant attribute removed.

[4]It might be (justifiably) argued here that we should have used a separate run with no pruning on all of the data for a comparison. Our main purpose, however, was to compare different pruning approaches and not evaluate the merits of pruning by itself. The results for the initial overfitting phases of REP, GROW and TDP may nevertheless be an indicator for the latter (and they come at no additional cost).

Summary	Table 1	Table 2	Table 3	Total
C4.5	83.63	86.40	92.24	87.42
Fossil	82.33	87.25	94.12	87.90
No Pruning	78.43	85.46	93.72	85.87
REP	81.49	86.77	93.22	87.16
Grow	80.81	86.25	93.05	86.70
No Pruning (TDP)	80.56	86.71	93.33	86.87
TDP	82.88	85.66	92.75	87.10
I-REP	82.34	86.10	92.29	86.91

Table 4: Average performances of the algorithms

Table 4 shows the average performances of the algorithms in each of the three different groups and a total comparison. FOSSIL seems to a good choice for all domains. I-REP's comparative performance seems to decrease when the noise level in the data decreases. This is probably due to its very strong *Overfitting Avoidance Bias*, which may lead to over-generalizations in noise-free domains. REP is better with lower noise levels for the opposite reason: Its specific-to-general search strategy may get stuck in a local optimum, which can lead to over-specializations some cases.

4 Conclusion

In terms of accuracy no clear recommendation for any of the above algorithms can be made. However, all of them seem to be competitive to classical propositional learners like C4.5. In terms of speed I-REP has proved to be the most efficient.

Pre-pruning as used in FOSSIL is the most stable method and should be used when nothing is known about the domain. A setting of 0.3 for the cutoff parameter has yielded good results in almost all tested domains, which illustrates the robustness of this algorithm against varying noise levels and varying training set sizes.

Reduced Error Pruning counterintuitively compares well to other approaches at low noise levels. The reason for this is that REP's specific-to-general search strategy may get stuck at too specific theories in particular at high noise levels, where the initial theory has to be pruned a lot.

The search for a good starting theory as performed in *Top-Down Pruning* leads to a stable behavior. It brings speed-up and a high accuracy at higher noise levels, while it will still yield good results at low noise levels, although its general-to-specific search strategy is inappropriate then and may lead to higher run-times.

The integration of pre- and post-pruning has confirmed its significant speed-up compared to other methods that include post-pruning. In noisy domains, where the problems of mere post-pruning become most apparent, *Incremental Reduced Error Pruning* has also exhibited a slight gain in accuracy. However, its strong bias for simple theories leads to over-generalization with decreasing noise levels. The clear superiority that I-REP exhibited in the relational domains of KRK endgames (figure 6) and finite element mesh design [Fürnkranz and Widmer, 1994] could not be confirmed in propostional domains. Maybe the problem with rule interactions in the separate-and-conquer strategy (see section 2.4) is less detrimental in propostional domains.

One reason for FOSSIL's stability might be that it was the only algorithm that was able to learn from *all* of the training data. We are currently thinking of improving the other pruning algorithms by evaluating theories based on Rissanen's Minimum Description Length Principle [Rissanen, 1978] instead of separating part of the training data for estimating accuracy. This problem could also be avoided by using cross-validation. In particular for the efficient I-REP the additional computational costs caused by this method might still be bearable.

Acknowledgements

This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)*. 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 my colleagues B. Pfahringer and Ch. Holzbaur for help on many improvements of the PROLOG implementation of FOIL and G. Widmer for many suggestions and discussions that improved the quality of this paper.

References

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, Pacific Grove, CA, 1984.

[Brunk and Pazzani, 1991] Clifford A. Brunk and Michael J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 389–393, Evanston, Illinois, 1991.

[Cameron-Jones and Quinlan, 1993] R.M. Cameron-Jones and J.R. Quinlan. First order learning, zeroth order data. In *Proceedings of the 6th Australian Joint Conference on AI.* World Scientific, 1993.

[Cestnik *et al.*, 1987] Bojan Cestnik, Igor Kononenko, and Ivan Bratko. ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In Ivan Bratko and Nada Lavrač, editors, *Progress in Machine Learning*, pages 31–45, Wilmslow, England, 1987. Sigma Press.

[Clark and Niblett, 1989] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

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

[Džeroski and Bratko, 1992] Sašo Džeroski and Ivan Bratko. Handling noise in Inductive Logic Programming. In *Proceedings of the International Workshop on Inductive Logic Programming*, Tokyo, Japan, 1992.

[Esposito *et al.*, 1993] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. Decision tree pruning as a search in the state space. In *Proceedings of the European Conference on Machine Learning*, pages 165–184, Vienna, Austria, 1993. Springer-Verlag.

[Fürnkranz and Widmer, 1994] Johannes Fürnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 70–77, 1994.

[Fürnkranz, 1994a] Johannes Fürnkranz. A comparison of pruning methods for relational concept learning. In *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 371–382, 1994.

[Fürnkranz, 1994b] Johannes Fürnkranz. FOSSIL: A robust relational learner. In *Proceedings of the European Conference on Machine Learning*, pages 122–137, Catania, Italy, 1994. Springer-Verlag.

[Fürnkranz, 1994c] Johannes Fürnkranz. Top-down pruning in relational learning. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 453–457, Amsterdam, The Netherlands, 1994.

[Holte, 1993] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[Mingers, 1989] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[Mittenecker, 1977] Erich Mittenecker. *Planung und statistische Auswertung von Experimenten*. Verlag Franz Deuticke, Vienna, Austria, 8th edition, 1977. In German.

[Muggleton *et al.*, 1989] Stephen Muggleton, Michael Bain, Jean Hayes-Michie, and Donald Michie. An experimental comparison of human and machine learning formalisms. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 113–118, 1989.

[Pagallo and Haussler, 1990] Giulia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.

[Quinlan, 1987] John Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[Quinlan, 1990] John Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Quinlan, 1993] John Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.