A Comparison of Pruning Methods for Relational Concept Learning

Johannes Fürnkranz

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

Abstract

Pre-Pruning and Post-Pruning are two standard methods of dealing with noise in concept learning. Pre-Pruning methods are very efficient, while Post-Pruning methods typically are more accurate, but much slower, because they have to generate an overly specific concept description first. We have experimented with a variety of pruning methods, including two new methods that try to combine and integrate pre- and post-pruning in order to achieve both accuracy and efficiency. This is verified with test series in a chess position classification task.

1 Introduction

Inductive Logic Programming (ILP) or *Relational Learning* has established itself as one of the major research areas in the field of Machine Learning [Muggleton, 1992, Lavrač and Džeroski, 1993]. The ability to define concepts from data distributed in separate relational tables makes ILP methods particularly appropriate for learning from relational databases (see e.g. [Džeroski and Lavrač, 1993]).

However, data from real-world problems often are *noisy* or incomplete. Learning algorithms that are meant to discover knowledge in real-world domains must be able to deal with this problem [Matheus *et al.*, 1993]. Another major issue for learning from real-world data is *efficiency* [Frawley *et al.*, 1992]. [Muggleton, 1993] argues that in many interesting real-world problems — such as the protein prediction problem [Muggleton *et al.*, 1992] on which the ILP system Golem [Muggleton and Feng, 1990] was tried — millions of examples may be available for training, which is beyond the scope of most ILP systems of today.

This paper is mainly concerned with achieving noise-tolerance through efficient *Pruning* methods for relational learners. We will give a short introduction to relational learning algorithms in section 2. Section 3 discusses several approaches to Pruning in ILP and presents two new approaches to combining and integrating the two basic methods — *Pre-Pruning* and *Post-Pruning*. In section 4 we give an experimental comparison of all the methods and then draw some conclusions (section 5).

2 Inductive Logic Programming

Inductive Logic Programming (ILP) can be viewed at research on the intersection of Logic Programming and Inductive Machine Learning [Muggleton, 1992]. In short the research concentrates on the induction of PROLOG programs from relational data. Being able to express the discovered knowledge in a first-order logic representation language can overcome some of the limitations of classical learning algorithms. Most of them learn from an attribute-value representation of the input data and their representational power thus is restricted to decision trees [Quinlan, 1983] or propositional Horn clauses [Michalski *et al.*, 1986, Clark and Niblett, 1989]. ILP algorithms on the other hand can not only test attributes for specific values, but also make use of relations between the different attributes. [Džeroski and Lavrač, 1993] discuss these issues.

The learning task for most relational learning systems is as follows:

Given:

- a set of positive and negative training examples
- background knowledge (typically in the form of relational tables)

Find:

- a logic program that is able to derive all of the positive and none of the negative training examples using the relations provided in the background knowledge.

Many ILP algorithms — including all of the algorithms discussed in this paper — address this problem with the so-called *separate-and-conquer* strategy used in FOIL [Quinlan, 1990]. The basic idea behind this approach is to learn one rule after the other until all of the positive examples are covered by at least one rule. The rules are constructed by evaluating each variabilization of each relation in the background knowledge with a greedy heuristic like information gain and then selecting the most promising candidate as the next literal of the clause. This process is repeated until enough conditions have been found to rule out all of the negative examples. All positive examples covered by this rule are then removed from the training set. Rules are learned in this way until no positive examples are left.

ILP systems have already been applied on various real-world problems. Some of the systems even produced new knowledge that was of considerable interest for researchers in the application domain and has been published in journals of their subject area [Muggleton *et al.*, 1992, Sternberg *et al.*, 1992, King *et al.*, 1992].

Two of the main problems when dealing with real-world data are noise-tolerance and efficiency. [Muggleton, 1993] argues that many real-world problems may involve thousands or even millions of examples, so that efficiency is a major issue for research in ILP. In the next section we will discuss several ways of achieving noise-tolerance in ILP programs and will evaluate them with respect to efficiency.

3 Pruning in Relational Learning Algorithms

Pruning is a standard way of dealing with noise in concept learning (see e.g. [Mingers, 1989] or [Esposito *et al.*, 1993]). There are two fundamentally different approaches [Cestnik *et al.*, 1987], *Pre-Pruning* and *Post-Pruning*. In sections 3.1 and 3.2 we will review some of these methods that have been adopted for relational concept learning systems. In section 3.3 we show how pre-pruning can be used for improving the efficiency and accuracy of post-pruning by providing a better starting theory. Finally, in section 3.4, we introduce a method that actually integrates both approaches by using post-pruning methods as a pre-pruning criterion.

3.1 Pre-Pruning

Pre-Pruning methods deal with noise during concept generation. Algorithms that have no pre-pruning heuristics are typically prone 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 they therefore learn overly specific clauses. Many of these rules mostly cover noisy examples and are bad generalizations. By learning a few "over-general" rules instead of many very specific clauses, several algorithms deliberately cover some negative training examples and leave some positive training examples uncovered in order to avoid the above problem.

In ILP, pre-pruning has been common in the form of *stopping criteria*, i.e. heuristics that determine when to stop adding conditions to a rule, or when to stop adding rules to the concept description. The most commonly used 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 a theory together with its exceptions will not exceed the costs of encoding the examples as they are.

- *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, 1994a] only adds a condition to a rule when its heuristic value is above a predefined threshold.

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, 1994a].

3.2 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 payed 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 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 in accuracy on the pruning set, in a first step the intermediate theory is augmented with generalizations of all its clauses. In a second step, clauses from this expanded theory are subsequently selected to form the final concept description until no further clause improves predictive accuracy on the pruning set. For noisy data the asymptotic costs of this pruning algorithm have been shown to be below the costs of the initial phase of overfitting.

3.3 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. [Cohen, 1993] therefore further improves the GROW algorithm by adding two weak MDL-based stopping criteria. These methods are not intended to entirely prevent overfitting like the pre-pruning approaches of section 3.1, 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.

The same is attempted in the *top-down pruning* (TDP) approach described in [Fürnkranz, 1994b]. Here FOSSIL's simple cutoff stopping criterion (see section 3.1) is exploited for a powerful algorithm that generates all theories learnable by FOSSIL with different settings of the cutoff parameter [Fürnkranz, 1994a]. From these the most specific theory within one standard error of classification of the most accurate theory is selected as the starting theory for the post-pruning phase. This is quite similar to 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. However, the implementation of TDP made use of several optimizations, so that finding this theory is often cheaper than fitting the noise. In particular, the theories are generated in a top-down fashion and are evaluated on the way, so that learning can stop after a theory has been found that is below one standard error of the best theory so far. A more detailed description of this process can be found in [Fürnkranz, 1994b].

3.4 Integrating Pre-and Post-Pruning

There are several problems with pruning in relational concept learning. Not all of them are attacked by the algorithms in the previous sections [Fürnkranz and Widmer, 1994]. In particular, the separate-and-conquer strategy used in all FOIL-like ILP systems (see section 2) 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. These negative instances are deliberately ignored, i.e. they are practically identified to be noisy. Consequently these 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 those instances 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 be selected. A wrong choice of a literal cannot be undone by pruning.

[Fürnkranz and Widmer, 1994] present I-REP, 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.

4 Experiments

We have tested several algorithms in the domain of recognizing illegal chess positions in the KRK chess endgame [Muggleton *et al.*, 1989]. This domain has become a standard benchmark problem for relational learning systems. The goal is to learn the concept of an illegal white-to-move position with only white king, white rook and black king on the board. The target predicate is `illegal(A,B,C,D,E,F)` where the parameters correspond to the row and file coordinates of the pieces in the above order. Background knowledge consists of the relations `X < Y`, `X = Y` and `adjacent(X,Y)`. This task would be very hard for attribute-value learning systems, which typically cannot make use of background knowledge in the form of relations (but see [Džeroski and Lavrač, 1993] for a way to circumvent this problem). The domain contains a total of 262,144 different instances; 86,976 (33.18%) of them are illegal.

```
illegal(A,B,C,D,E,F) :- C = E.
illegal(A,B,C,D,E,F) :- D = F.
illegal(A,B,C,D,E,F) :- adjacent(A,E), adjacent(B,F).
illegal(A,B,C,D,E,F) :- A = C, B = D.
```

Figure 1: An approximate KRK theory that is 99.57% correct.

The signs of 10% of the training instances were deliberately reversed to generate noise in the data. The learned concepts were evaluated on test sets with 5000 noise-free examples. The sets used were the same as in [Fürnkranz, 1994a]. See also the Appendix of [Fürnkranz, 1993] for a closer description of the domain.

The tested algorithms were

- the pre-pruning systems FOIL 6.1 [Quinlan and Cameron-Jones, 1993] and FOSSIL (with a cutoff of 0.3) [Fürnkranz, 1994a],
- the post-pruning systems REP [Brunk and Pazzani, 1991] and GROW [Cohen, 1993],
- the combined system TDP [Fürnkranz, 1994b],
- and the integrated system I-REP [Fürnkranz and Widmer, 1994].

All algorithms were implemented by the author in PROLOG except for FOIL 6.1 which is written in C and publicly available from `ftp.cs.su.oz.au`. The PROLOG systems had major parts of their implementations in common. In particular they shared the same interface to the data. GROW used the same pruning operators as REP[1]. FOIL 6.1 was used with its default settings except that the `-V 0` option was set to avoid the introduction of new variables. The PROLOG systems had all of their argument modes declared as input, which has the same effect. The only difference in search space between the PROLOG systems and FOIL 6.1 was that the former did not consider recursive literals for efficiency reasons. FOIL 6.1 would probably have been faster if this had been enforced, but no significant difference in accuracy can be expected (see [Fürnkranz, 1994a] for results from experiments with the same data sets, where the code of FOIL 4 was modified to prevent recursion). Run-times for most algorithms were measured in CPU seconds for SUN Sparc stations ELC. The experiments with FOSSIL, FOIL 6.1 and TDP, however, were performed on a SUN Sparc station IPX, which gave the other algorithms a 5 : 6 advantage (determined on a few test runs on both machines).

The algorithms were trained on identical sets of sizes from 100 to 1000 examples. All reported results were averaged over 10 runs, except for the training set size 1000, where only 6 runs were performed, because of the complexity of this task for some algorithms.

Figure 2 shows curves for accuracy and run-times over 5 different training set sizes. I-REP — after a bad start with only 84.55% accuracy on 100 examples — achieves the

[1] In [Cohen, 1993] REP and GROW also used identical pruning operators. They were, however, slightly different from our choice. We used the operators described in [Brunk and Pazzani, 1991].

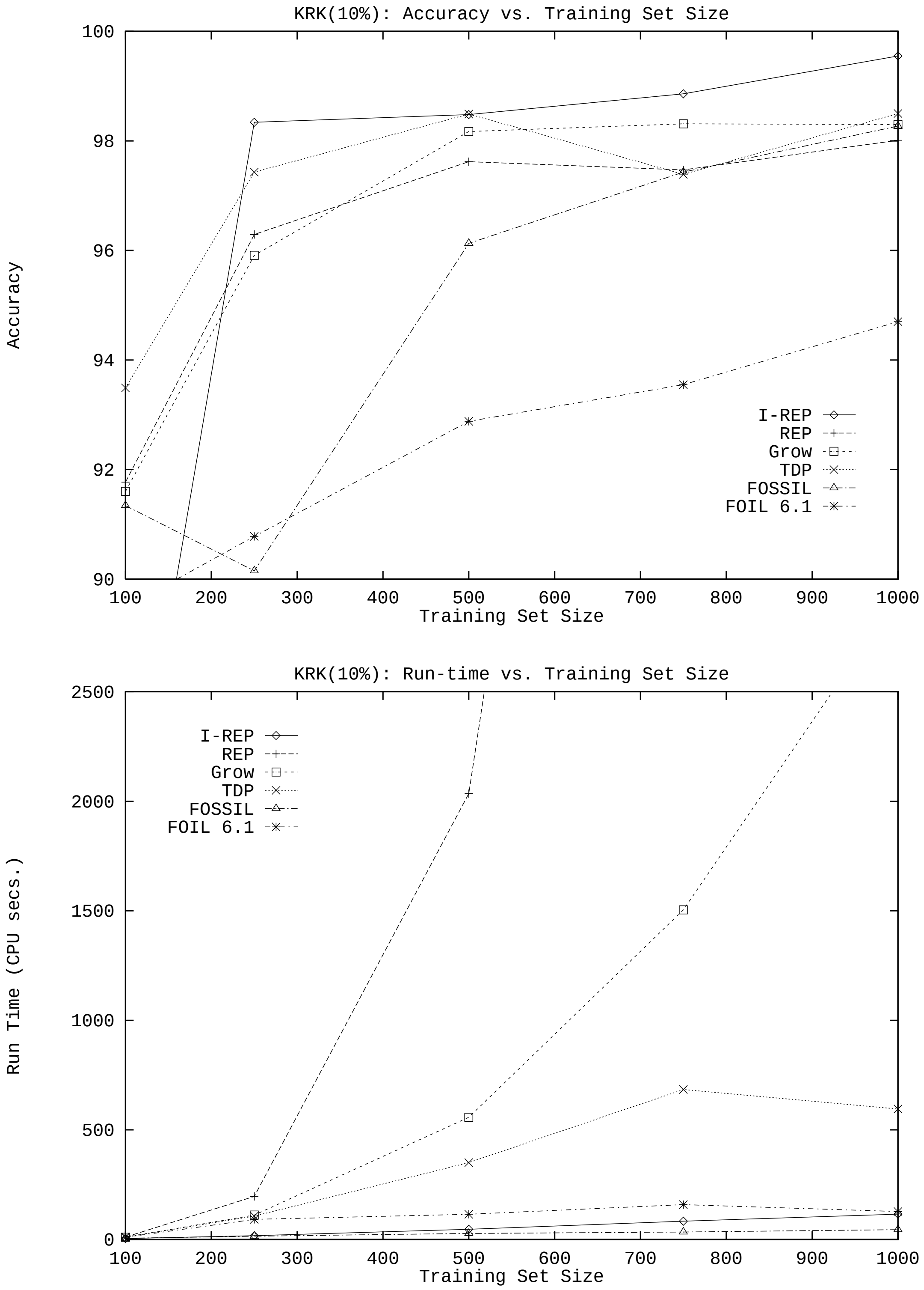

Figure 2: KRK domain (10% noise), different training set sizes

highest accuracy. In predictive accuracy, FOIL did poorly. Its stopping criterion is dependent on the training set size and thus too weak to effectively prevent overfitting the noise [Fürnkranz, 1994a]. With 1000 examples FOIL learns concepts that have more than 20 rules and are incomprehensible. I-REP on the other hand consistently produces the 99.57% correct, understandable 4-rule approximation of the correct concept description of figure 1. This theory correctly identifies all illegal positions, except the ones where the white king is between the black king and the white rook and thus blocks a check that would make the position illegal, because white is to move. The post-pruning approaches REP and Grow both are about equal, and TDP does not lose accuracy compared to them. All three, however, only rarely find the 4th rule of figure 1. It can also be seen that the pre-pruning approach taken by FOSSIL needs many examples in order to make its heuristic pruning decisions more reliable.

	Accuracy		Run-time		
Algorithm	Pre	Post	Pre	Post	Total
FOIL 6.1	**94.70**	—	127.17	—	**127.17**
FOSSIL (0.3)	**98.27**	—	44.39	—	**44.39**
REP	85.65	**98.01**	2129.89	23125.34	**25255.23**
Grow	85.65	**98.30**	2129.89	806.89	**2936.78**
TDP	96.56	**98.50**	433.54	162.25	**595.79**
I-REP	—	**99.55**	—	—	**115.35**

Table 1: KRK domain (10% noise), 1000 examples

FOSSIL, on the other hand, is the fastest algorithm. FOIL, although implemented in C, is slower, because with increasing training set sizes it learns more clauses than FOSSIL [Fürnkranz, 1994a] and has extensive backtracking mechanisms. REP proves that its pruning method is very inefficient. Grow has an efficient pruning algorithm, but still suffers from the expensive overfitting phase. TDP brings a little speed-up compared with REP and Grow, which is mainly due to the fact that it is able to start post-pruning with a much better theory than REP or Grow, as can be seen from table 1. I-REP, however, learns a much better theory and is faster than both, the growing and the pruning phase of TDP.

In fact, I-REP, where post-pruning is integrated into a pre-pruning criterion, is only a little slower than FOSSIL, but much more accurate. Thus it can be said that it truly combines the merits of post-pruning (accuracy) and pre-pruning (efficiency). This becomes also apparent in figure 3, where accuracy (with the standard deviations observed in the different runs) is plotted against the logarithm of the run-time.

I-REP has also been tested on some of the standard Machine Learning databases that can be found in the UCI repository (`ics.uci.edu`). and again was significantly faster than REP or Grow [Fürnkranz and Widmer, 1994]. On most natural domains (like *Mesh, Promoters* and *Votes*) it achieved a higher accuracy than REP and Grow [Fürnkranz and Widmer, 1994]. However, it did worse than both of them in domains with low redundancy, where rules have to be found that cover only a few training examples (e.g.

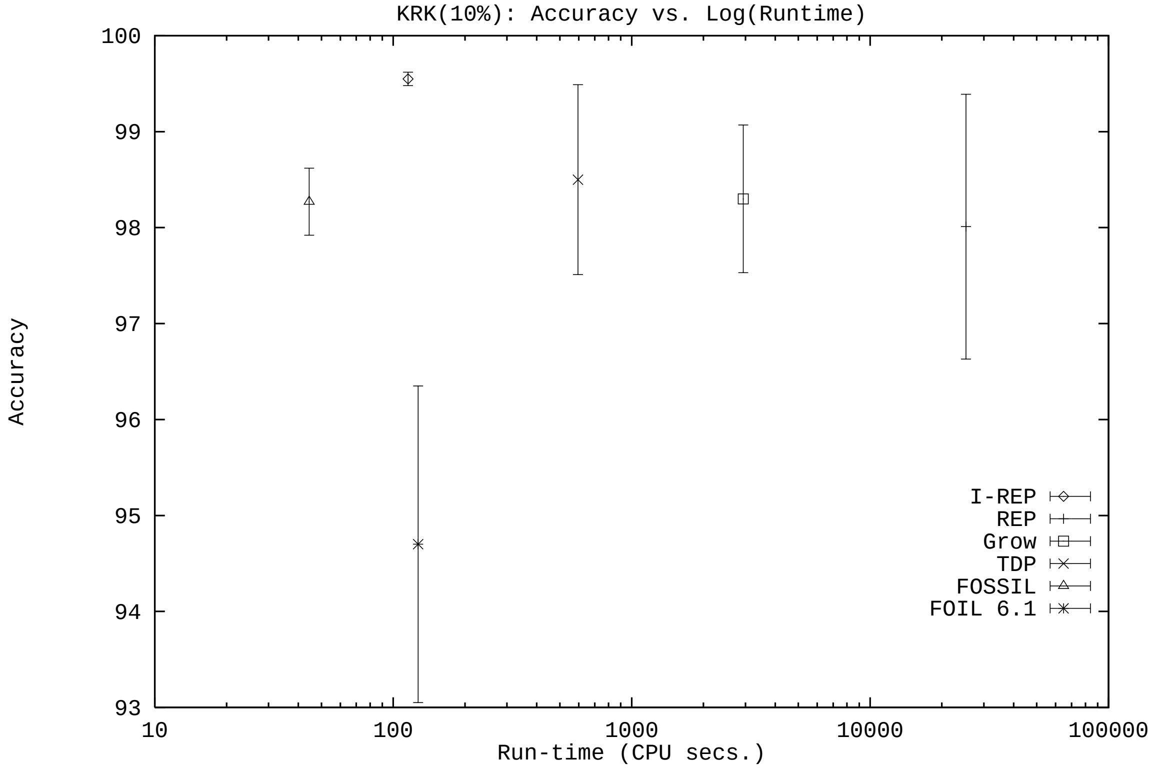

Figure 3: KRK domain (10% noise), 1000 examples

Tic-Tac-Toe). In these domains post-pruning methods generally do not do very well, as they all tend to over-generalize. I-REP, however, does more so than the other algorithms.

5 Conclusion

We have compared several approaches to pruning in relational concept learning. Conventional pre-pruning methods are very efficient, but not always as accurate as post-pruning methods. The latter, however, tend to be very expensive, because they have to learn an over-specialized theory first. I-REP [Fürnkranz, 1994a] is an algorithm that integrates pre- and post-pruning into one criterion. Our experiments show that this approach effectively combines the efficiency of pre-pruning with the accuracy of post-pruning in domains with high redundancy. As real-world databases typically are large and noisy [Matheus *et al.*, 1993], and thus require learning algorithms that are both efficient and noise-tolerant, I-REP seems to be an appropriate choice for this purpose.

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.

[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.

[Džeroski and Lavrač, 1993] Sašo Džeroski and Nada Lavrač. Inductive learning in deductive databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):939–949, December 1993. Special Issue on Learning and Discovery in Knowledge-Based Databases.

[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.

[Frawley *et al.*, 1992] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, pages 57–70, Fall 1992. Also in [Piatetsky-Shapiro and Frawley, 1991].

[Fürnkranz and Widmer, 1994] Johannes Fürnkranz and Gerhard Widmer. Incremental Reduced Error Pruning. In *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, 1994.

[Fürnkranz, 1993] Johannes Fürnkranz. FOSSIL: A robust relational learner. Technical Report OEFAI-TR-93-28, Austrian Research Institute for Artificial Intelligence, 1993. Extended version.

[Fürnkranz, 1994a] 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, 1994b] 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.

[King *et al.*, 1992] R. King, S. Muggleton, R. Lewis, and M. Sternberg. Drug design by Machine Learning: The use of Inductive Logic Programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23), 1992.

[Lavrač and Džeroski, 1993] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1993.

[Matheus *et al.*, 1993] Christopher J. Matheus, Philip K. Chan, and Gregory Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):903–913, December 1993.

[Michalski *et al.*, 1986] R. S. Michalski, I. Mozetič, J. Hong, and N. Lavrač. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, 1986.

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

[Muggleton and Feng, 1990] Stephen H. Muggleton and Cao Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 1–14, Tokyo, Japan, 1990.

[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.

[Muggleton *et al.*, 1992] S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based Machine Learning. *Protein Engineering*, 5(7):647–657, 1992.

[Muggleton, 1992] Stephen Muggleton, editor. *Inductive Logic Programming.* Academic Press Ltd., London, 1992.

[Muggleton, 1993] Stephen H. Muggleton. Inductive Logic Programming: Derivations, successes and shortcomings. In Pavel B. Brazdil, editor, *Machine Learning: ECML-93*, number 667 in Lecture Notes in Artificial Intelligence, pages 21–37, Vienna, Austria, 1993. Springer-Verlag.

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

[Piatetsky-Shapiro and Frawley, 1991] Gregory Piatetsky-Shapiro and William J. Frawley, editors. *Knowledge Discovery in Databases.* MIT Press, 1991.

[Quinlan and Cameron-Jones, 1993] John Ross Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20, Vienna, Austria, 1993.

[Quinlan, 1983] J. Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell, editors, *Machine Learning. An Artificial Intelligence Approach*, pages 463–482. Tioga Publishing Co., 1983.

[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.

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

[Sternberg *et al.*, 1992] M. Sternberg, R. Lewis, R. King, and S. Muggleton. Modelling the structure and function of enzymes by machine learning. *Proceedings of the Royal Society of Chemistry: Faraday Discussions*, 93:269–280, 1992.