A Tight Integration of Pruning and Learning (Extended Abstract)

Johannes Fürnkranz

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna
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.

1 Introduction

Most rule learning algorithms deal with noise in the data during learning, i.e. they employ *pre-pruning*. In relational learning systems like FOIL [Quinlan, 1990] 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.

2 REP

The most common post-pruning algorithm, *Reduced Error Pruning (REP)*, has been adopted from propositional decision tree learning to relational rule learning [Brunk and Pazzani, 1991]. 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 or the relational learner FOIL. 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 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. Although the two approaches share many similarities, 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. One 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 to prune the nodes bottom up, i.e. only allow to delete clauses from the end of the program. REP, however, 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 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.

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 Sect. 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.

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 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.

4 I^2-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.

As in pre-pruning algorithms I^2-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. 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 Results

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. The results can be found in [Fürnkranz, 1995] which is available via anonymous `ftp` from `ftp.ai.univie.ac.at`. I-REP and I^2-REP are both significantly faster than REP. In the KRK domain they also learn significantly better theories, in particular at high training set sizes. In addition, I^2-REP inproves upon I-REP on small training set sizes. The price that has to be paid for this is that I^2-REP is a little slower than I-REP. Nevertheless the experiments showed that both algorithms have about the same subquadratic asymptotic time complexity, while REP's asymptotic time complexity has been confirmed to be $\Omega(n^4)$.

Acknowledgements

This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P10489-MAT. 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 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.

[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.

[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, New Brunswick, NJ, 1994.

[Fürnkranz, 1995] Johannes Fürnkranz. A tight integration of pruning and learning. Technical Report OEFAI-TR-95-03, Austrian Research Institute for Artificial Intelligence, 1995.

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

[Weiss and Indurkhya, 1994] Sholom M. Weiss and Nitin Indurkhya. Small sample decision tree pruning. In *Proceedings of the 11th Conference on Machine Learning*, pages 335–342, Rutgers University, New Brunswick, NJ, 1994.