Appears in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp.1135-1140, Chambery, France, 1993

Symbolic Revision of Theories with M-of-N Rules *

Paul T. Baffes and Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, TX 78712
baffes@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

This paper presents a major revision of the EITHER propositional theory refinement system. Two issues are discussed. First, we show how run time efficiency can be greatly improved by changing from a exhaustive scheme for computing repairs to an iterative greedy method. Second, we show how to extend EITHER to refine M-of-N rules. The resulting algorithm, NEITHER (New EITHER), is more than an order of magnitude faster and produces significantly more accurate results with theories that fit the M-of-N format. To demonstrate the advantages of NEITHER, we present preliminary experimental results comparing it to EITHER and various other systems on refining the DNA promoter domain theory.

1 Introduction

Recently, a number of machine learning systems have been developed that use examples to revise an approximate (incomplete and/or incorrect) domain theory [Ginsberg, 1990; Ourston and Mooney, 1990; Towell and Shavlik, 1991; Danyluk, 1991; Whitehall *et al.*, 1991; Matwin and Plante, 1991]. Most of these systems revise theories composed of strict if-then rules (Horn clauses). However, many concepts are best represented using some form of partial matching or evidence summing, such as M-of-N concepts, which are true if at least M of a set of N specified features are present in an example. There has been some work on the induction of M-of-N rules that demonstrates the advantages of this representation [Spackman, 1988; Murphy and Pazzani, 1991]. Other work has focused on revising rules that have real-valued weights [Towell and Shavlik, 1992; Mahoney and Mooney, 1992]. However, revising theories with simple M-of-N rules has not previously been addressed. Since M-of-N rules are more constrained than rules with real-valued weights, they provide a stronger bias and are easier to comprehend.

This paper presents a major revision of the EITHER propositional theory refinement system [Ourston and Mooney, 1990; Ourston and Mooney, in press] that is significantly more efficient and is also capable of revising theories with M-of-N rules. EITHER is inefficient because it computes a potentially exponential number of repairs for each failing example. The new version, NEITHER (New EITHER), computes only the single best repair for example, and is therefore much more efficient.

Also, because it was restricted to strict Horn-clause theories, EITHER did not produce as accurate results as KBANN (a neural-network revision system) on the DNA promoter problem [Towell and Shavlik, 1991; Towell and Shavlik, 1992]. Some aspects of the promoter concept fit the M-of-N format, since there are several potential sites where hydrogen bonds can form between the DNA and a protein; if enough of these bonds form, promoter activity can occur. EITHER attempts to learn this concept by forming a separate rule for each potential configuration by deleting different combinations of antecedents from the initial rules. Since a combinatoric number of such rules is needed to accurately model an M-of-N concept, the generality of the resulting theory is impaired. NEITHER, however, includes the ability to generalize a rule by lowering the threshold on an M-of-N rule. Including threshold changes as an alternative method for covering misclassified examples was easily incorporated within the basic EITHER framework.

To demonstrate the advantages of NEITHER, we present experimental results comparing it to EITHER and various other systems on refining the promoter domain theory. NEITHER runs more than an order of magnitude faster than EITHER and produces a significantly more accurate theory with minor revisions that are easy to understand.

2 Theory Revision Algorithm

2.1 The EITHER Algorithm

The original EITHER theory refinement algorithm has been presented in various levels of detail in [Ourston and Mooney, 1990; Ourston and Mooney, in press; Ourston, 1991]. It was designed to repair propositional Horn-clause theories that are either overly-general or

*This research was supported by the NASA Graduate Student Researchers Program under grant number NGT-50732, the National Science Foundation under grant IRI-9102926, and a grant from the Texas Advanced Research Program under grant 003658144.

overly-specific or both. An overly-general theory is one that causes an example (called a *failing negative*) to be classified in categories other than its own. EITHER specializes existing antecedents, adds new antecedents, and retracts rules to fix these problems. An overly-specific theory causes an example (called a *failing positive*) not to be classified in its own category. EITHER retracts and generalizes existing antecedents and learns new rules to fix these problems. Unlike other theory revision systems that perform hill-climbing (and are therefore subject to local maxima), EITHER is guaranteed to fix any arbitrarily incorrect propositional Horn-clause theory [Ourston, 1991].

EITHER Main Loop

```
Compute all repairs for each example
While some examples remain uncovered
    Add best repair to cover set
    Remove examples covered by repair
end
Apply repairs in cover set to theory
```

NEITHER Main Loop

```
While some examples remain
    Compute a single repair for each example
    Apply best repair to theory
    Remove examples fixed by repair
end
```

Figure 1: Comparison of EITHER and NEITHER algorithms.

The algorithm used by EITHER for both generalization and specialization is shown in the top half of Figure 1. There are three basic steps. First, *all* possible repairs for each failing example are computed. Next, EITHER enters a loop to compute a subset of these repairs that can be applied to the theory to fix all of the failing examples. This subset is called a *cover*. Repairs are ranked according to a benefit-to-cost ratio that trades off the number of examples covered against the size of the repair and the number of new failing examples it creates. The best repair is added to the cover on each iteration. Lastly, the repairs in the cover are applied to the theory. If the application of a repair over-compensates by creating new failing examples, EITHER passes the covered examples and the new failing examples to an induction component.[1] The results of the induction are added as a new rule when generalizing or as additional antecedents when specializing.

The time consuming part of this algorithm is the first step where all repairs for a given failing example are found. Figure 2 illustrates this process for theory generalization where EITHER is searching for leaf-rule[2] antecedent retractions to correct failing positive examples. The upper half of the diagram shows an input theory both as rules (on the left) and as an AND-OR graph. The lower half of the diagram shows a hypothetical failing positive example and its partial proofs.[3] From these proofs there are four possible repairs which will fix the example: retract `h,j,m`; retract `h,j,o,p`; retract `k,m`; retract `k,o,p`. Theory specialization follows a similar process to return sets of leaf-rule retractions which fix individual failing negative examples.

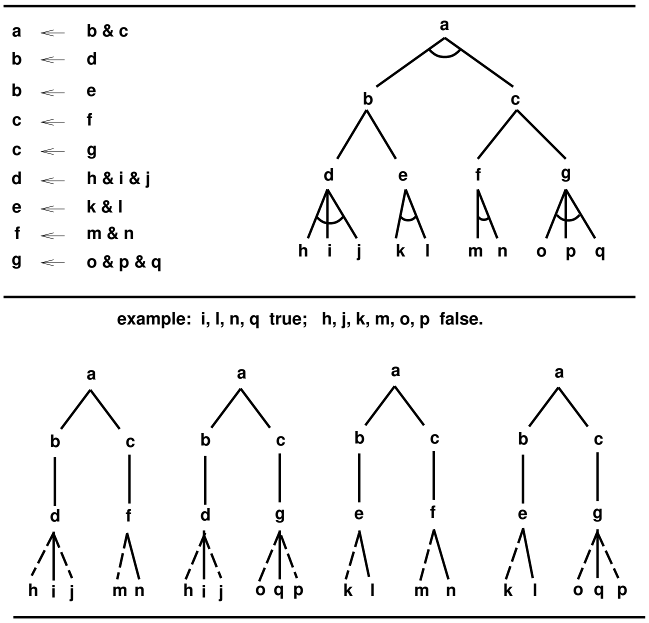

Figure 2: Partial proofs for unprovable positive example. Unprovable antecedents are shown with dotted lines.

2.2 Speeding Up EITHER

We have recently implemented a new version of EITHER (NEITHER) that takes a different approach, as shown in the bottom half of Figure 1. Two new algorithms form the basis for the difference between EITHER and NEITHER. First, calculation of repairs is now achieved in linear time. Second, all searches through the theory (for deduction, antecedent retraction and rule retraction) are optimized in NEITHER to operate in linear time by marking the theory to avoid redundant subproofs. NEITHER abandons the notion of searching for all partial proofs in favor of a greedy approach which rapidly selects a *single* best repair for each example. The three steps of the old EITHER algorithm can then be integrated into a single loop (see Figure 1).

To illustrate how repairs are computed in linear time, refer again to Figure 2. Rather than computing all partial proofs, NEITHER works bottom-up, constructing a single set of retractions. When multiple options exist, NEITHER alternates between returning the smallest option and returning the union of the options, depending whether the choice involves an AND or OR node. For

[1] EITHER uses a version of ID3 [Quinlan, 1986] for its induction.

[2] A leaf rule is a rule whose antecedents include an observable or an intermediate concept that is not the consequent of any existing rule.

[3] A partial proof is one in which some antecedents cannot be satisfied.

Generalization					Specialization				
change	*resulting rule*	`b`	`c`	`bc`	*change*	*resulting rule*	`b`	`c`	`bc`
orig. rule	`a` $\leftarrow$ `2 of (b,c)`	N	N	Y	orig. rule	`a` $\leftarrow$ `1 of (b,c)`	Y	Y	Y
threshold -1	`a` $\leftarrow$ `1 of (b,c)`	Y	Y	Y	threshold +1	`a` $\leftarrow$ `2 of (b,c)`	N	N	Y
delete `b`	`a` $\leftarrow$ `c`	N	Y	Y	delete rule	none	N	N	N

Table 1: Comparison of Revisions.

generalization, retractions are unioned at AND nodes because all unprovable antecedents must be removed to make the rule provable. At OR nodes, only the smallest set of retractions is kept since only one rule need be provable. For specialization, these choices are reversed. Results are unioned at OR nodes to disable all rules which fire for a faulty concept. At AND nodes, the smallest set of rule retractions is selected since any single failure will disable a rule.

As an example, in Figure 2 the antecedent retraction calculations for the example would begin at the root of the graph, recursively calling nodes `b` and `c`. Retraction for node `b` then recurses on nodes `d` and `e`. When the recursion returns back to node `b` a choice must be made between the results from nodes `d` and `e` because node `b` is an OR node. Since the latter requires fewer retractions, it is chosen as the return value for node `b`. This process continues, resulting in a final repair: retract `k,m`.

Note that this algorithm is linear in the size of the theory. No node is visited more than once, and the computation for choosing among potential retractions must traverse the length of each rule at most once. The final repair is also minimum with respect to the various choices made along the way; it is not possible to find a smaller repair that will satisfy the example. This new algorithm thus trades the complete information available in the partial proofs for speed in computation.

2.3 Adding M-of-N Rules to NEITHER

With M-of-N rules, there are six types of revisions that can be made to a theory. As before, antecedents may be deleted or rules may be added to generalize the theory, and antecedents may be added or rules deleted to specialize the theory. The two new revisions are to increase or decrease the threshold: decreasing generalizes a rule and increasing specializes it.

To incorporate these two new revisions, NEITHER must be changed in four places. First, the computation of a repair for each failing example must take thresholds into account. For generalization, one need only retract enough antecedents to make the rule provable; there is no need to retract all false antecedents if the rule has a threshold. For example, if the rule for `e` in Figure 2 had a threshold of 1 there would be no need to retract `k` to prove this rule. A similar accounting for thresholds is required for computing rule deletions for specialization. Note that during generalization the threshold of each rule from which antecedents are retracted must be decreased by the number of antecedents retracted to account for the smaller size of the rule.

Second, NEITHER must compute threshold repairs. Calculating threshold changes can be done in conjunction with the computation of antecedent and rule deletion repairs since it is directly related to how many of antecedents of a rule are provable. For generalization, we change the threshold to the number of antecedents which are provable. In specialization, we set the threshold to one more than the number of provable antecedents.

Third, a mechanism must be provided for selecting between a threshold change and a deletion. Effectively, this amounts to deciding which type of revision to try first. The philosophy used in NEITHER is to try the most aggressive changes initially in the hopes that the resulting repair will cover more examples. If the repair creates new failing examples, the less ambitious repairs are tried in turn with induction used as a last resort. During generalization, more radical repairs are those which create more general rules (i.e., rules which can prove more examples). In specialization, the opposite is true. As with EITHER, if all changes result in new failing examples, the algorithm falls back to induction to learn new rules or add new antecedents.

Table 1 compares equivalent threshold and deletion changes for generalization and specialization. The columns labeled with `b`, `c` and `bc` indicate whether the corresponding rule will conclude `a` when just `b`, just `c` or both `b` and `c` are true. Note that in both cases, the threshold change results in a more general rule. This means that threshold changes should be tried before antecedent deletions during generalization, but tried after rule deletions during specialization.

Fourth and finally, the induction component of NEITHER must be altered slightly to accommodate threshold rules. When the application of a repair causes new failing examples to occur, NEITHER resorts to induction as did EITHER. The result of the induction cannot, however, simply be added to the theory as before. Table 2 illustrates the problem. The original rule shown can be used to prove both the positive and negative examples, and deleting this rule or incrementing its threshold only prevents the positive example from being proved. Assume that induction returns a new feature, `d`, which can be used to distinguish the two examples (i.e., `d` is true for the positive example but false for the negative example). Because the original rule has a threshold, adding `d` directly will still allow both examples to prove the rule. This problem remains even if one tries to increment the threshold in addition to adding `d`. Instead, the rule must be *split* by renaming the consequent of the original rule, and creating a new rule with the renamed consequent and the results of induction as the new rule's antecedent list.

example features	*pos. example*	*neg. example*
b, c, d	b, ¬c, d	b, c, ¬d
orig. rule	*pos. example*	*neg. example*
a ← 1 of (b,c)	Y	Y
add to rule	*pos. example*	*neg. example*
a ← 1 of (b,c,d)	Y	Y
split rule	*pos. example*	*neg. example*
X ← 1 of (b,c)	Y	Y
a ← X,d	Y	N

Table 2: Induced Antecedent Addition.

3 Experimental Results

3.1 Experimental Design

For the purposes of this paper, the resulting algorithm is labeled NEITHER-MOFN. We tested both NEITHER and NEITHER-MOFN against other classification algorithms using the DNA promoter sequences data set [Towell *et al.*, 1990]. This data set involves 57 features, 106 examples, and 2 categories. The theory provided with the data set has an initial classification accuracy of 50%. We selected this particular data set because EITHER performed poorly on data sets best modelled using M-of-N rules. In addition to testing EITHER, NEITHER and NEITHER-MOFN, we ran experiments using ID3 [Quinlan, 1986], backpropagation [Rumelhart *et al.*, 1986] and RAPTURE [Mahoney and Mooney, in press] (a revision system based on certainty factors).

The experiments proceeded as follows. Each data set was divided into training and test sets. Training sets were further divided into subsets, so that the algorithms could be evaluated with varying amounts of training data. After training, each system's accuracy was recorded on the test set. To reduce statistical fluctuations, the results of this process of dividing the examples, training, and testing were averaged over 25 runs. The random seeds for the backpropagation algorithm were reset for each run. Training time, and test set accuracy were recorded for each run. Statistical significance was measured using a Student t-test for paired difference of means at the 0.05 level of confidence (i.e., 95% certainty that the differences are not due to random chance).

3.2 Results

The results of our experiments are shown in the three graphs of Figures 3, 4 and 5. Figure 3 compares the learning curves of the systems tested, showing how predictive accuracy on the test set changes as a function of the number of training examples. As can be seen NEITHER-MOFN's performance was significantly better than all other systems except RAPTURE and KBANN.[4] RAPTURE out-performed NEITHER-MOFN with small numbers of training examples but their accuracy was comparable with larger inputs. NEITHER's accuracy was on par with backpropagation, but was lower than EITHER for small training sets and higher than EITHER for large training sets. Note, that Figure 3 is not direct comparison of NEITHER and KBANN since the results reported were compiled from different subsets of the DNA promoter sequences data set. ID3 had significantly lower accuracy than the other systems.

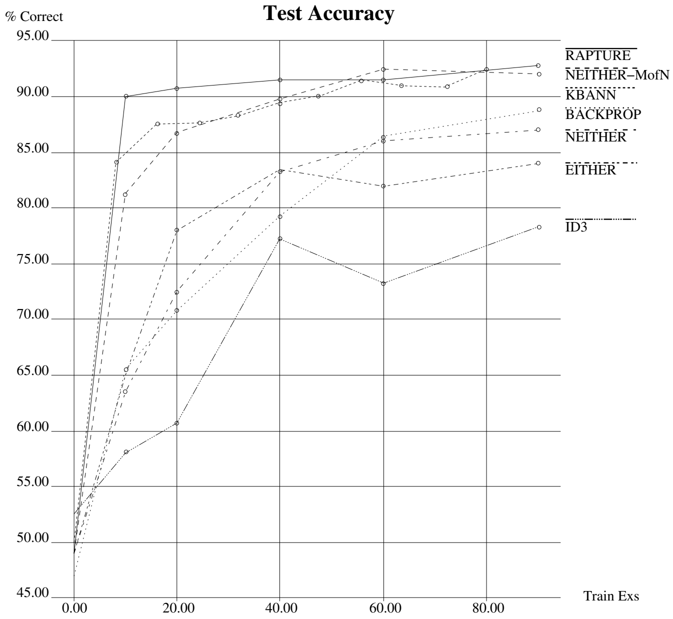

Figure 3: Test Set Accuracy

Figure 4 shows a comparison of training times. Both NEITHER-MOFN and NEITHER were more than an order of magnitude faster than backpropagation and EITHER. Only ID3 ran faster than NEITHER-MOFN.

We also collected data on the average complexity of the revised theories produced by both NEITHER and NEITHER-MOFN. Complexity was measured as the total size; i.e., the total number of all literals in the theory. The results are shown in Figure 5. As can be seen from this graph, NEITHER-MOFN not only produces less complex resulting theories but also produces theories closer in size to the original.

3.3 Discussion

Many of our expectations were borne out by the experimental results. Both NEITHER and NEITHER-MOFN ran more than an order of magnitude faster than EITHER due to the optimized algorithms discussed in section 2. NEITHER-MOFN's increase in accuracy was also expected since the new algorithm is able to concentrate on making M-of-N revisions directly. Also, the fact that NEITHER-MOFN generates less complex theories is not surprising, again because it can directly modify threshold values rather than create new rules. In short, by adding one more operator to the generalization and specialization processes, NEITHER-MOFN is able to accurately revise a theory known to be difficult for symbolic systems, without having to sacrifice the efficiency of a symbolic approach. Finally, the most comparable learning-curve results from [Towell, 1991] would indicate that KBANN's accuracy in the promoter domain is about

[4] Technically, the last difference between backpropagation and NEITHER-MOFN was only significant at the 0.1 level.

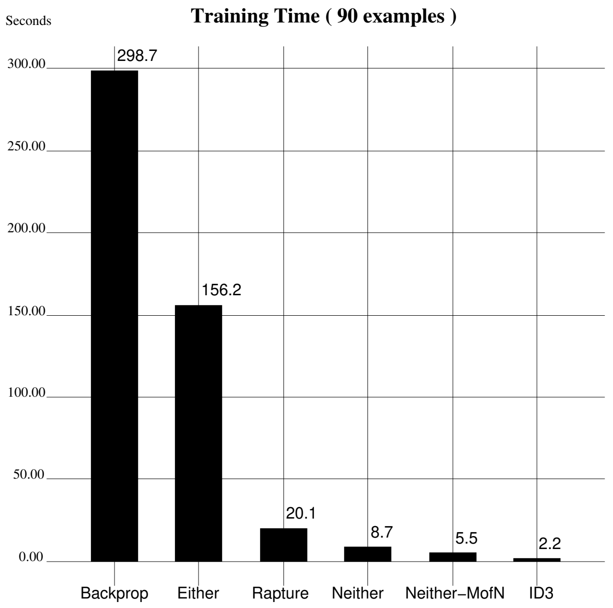

Figure 4: Training Time Comparison

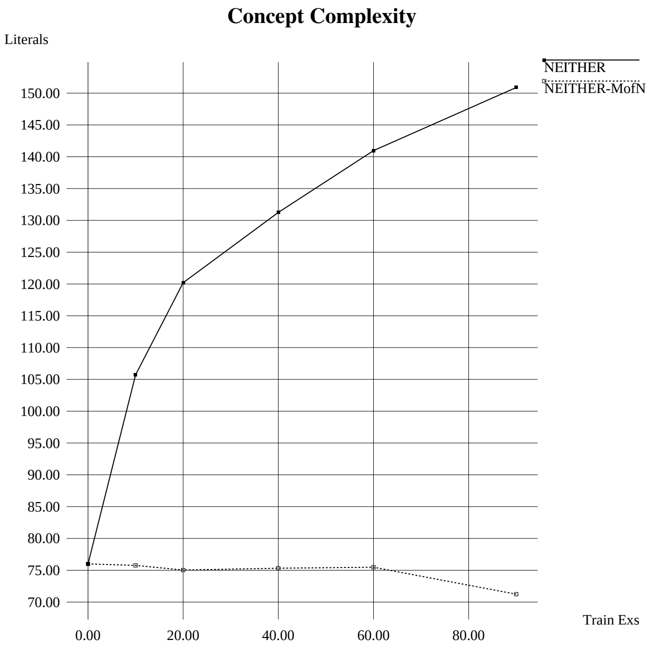

Figure 5: Concept Complexity

the same as NEITHER-MOFN's.

4 Related Work

Several researchers have developed methods for inducing M-of-N concepts from scratch. CRLS [Spackman, 1988] learns M-of-N rules and out-performed standard rule induction in several medical domains. ID-2-of-3 [Murphy and Pazzani, 1991] incorporates M-of-N tests in decision-tree learning and out-performed standard decision-tree induction in a number of domains. Both projects clearly demonstrate the advantages of M-of-N rules.

SEEK2 [Ginsberg *et al.*, 1988] includes operators for refining M-of-N rules; however, its revision process is heuristic and it is not guaranteed to produce a revised theory that is consistent with all of the training examples. NEITHER uses a greedy covering approach to guarantee that it finds a set of revisions that fix all of the misclassified examples in the training set. Also, unlike NEITHER, SEEK2 cannot learn new rules or add new antecedents to existing rules.

KBANN [Towell and Shavlik, 1992] revises a theory by translating it into a neural network, using backpropagation to refine the weights, and then retranslating the result back into symbolic rules. NEITHER's symbolic revision process is much more direct and, from all indications, significantly faster. Although KBANN's results are referred to as M-of-N rules, they actually contain real-valued antecedent weights and therefore are not strictly M-of-N. In addition, KBANN's revised theories for the promoter problem are also more complex in terms of number of antecedents than the initial theory [Towell, 1991], while NEITHER actually produces a slight reduction. Therefore, NEITHER's revised theories are less complex and presumably easier to understand. Finally, unlike KBANN, NEITHER is guaranteed to converge to 100% accuracy on the training data.

RAPTURE [Mahoney and Mooney, 1992] uses a combination of symbolic and neural-network learning methods to revise a certainty-factor rulebase [Buchanan and E.H. Shortliffe, 1984]. Consequently, it lies somewhere between NEITHER and KBANN on the symbolic-connectionist dimension. As illustrated in the results, its accuracy on the promoter problem is only slightly superior to NEITHER's. However, its real-valued certainty factors make its rules more complex.

5 Future Work

The current version of NEITHER needs to be enhanced to handle a number of issues. We need to incorporate a number of advanced features from EITHER, such as constructive induction, modification of higher-level rules, and the ability to handle numerical features and noisy data. Also, we could to extend our methods to handle negation as failure and incorporate the ability to handle M-of-N rules into first-order theory revision [Richards and Mooney, 1991]. Finally, we need to perform a more comprehensive experimental evaluation of the system.

6 Conclusions

This paper has presented an efficient propositional theory refinement system that is capable of revising M-of-N rules. The basic framework is a modification of EITHER [Ourston and Mooney, 1990]; however, the construction of partial proofs has been reduced from exponential to linear time and a method for revising the thresholds of M-of-N rules has been incorporated. The resulting system runs more than an order of magnitude faster and produces significantly more accurate results in domains requiring partial matching, such as the problem of recognizing promoters in DNA.

Acknowledgements

Special thanks to Chris Whatley for his help implementing NEITHER.

References

[Buchanan and E.H. Shortliffe, 1984] G.G. Buchanan and eds. E.H. Shortliffe. *Rule-Based Expert Systems:The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley Publishing Co., Reading, MA, 1984.

[Danyluk, 1991] A. D. Danyluk. Gemini: An integration of analytical and empirical learning. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 191–206, Harper's Ferry, W.Va., Nov. 1991.

[Ginsberg *et al.*, 1988] A. Ginsberg, S. M. Weiss, and P. Politakis. Automatic knowledge based refinement for classification systems. *Artificial Intelligence*, 35:197–226, 1988.

[Ginsberg, 1990] A. Ginsberg. Theory reduction, theory revision, and retranslation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 777–782, Detroit, MI, July 1990.

[Mahoney and Mooney, 1992] J. Mahoney and R. Mooney. Combining symbolic and neural learning to revise probabilistic theories. In *Proceedings of the ML92 Workshop on Integrated Learning in Real Domains*, Aberdeen, Scotland, July 1992.

[Mahoney and Mooney, in press] J. J. Mahoney and R. J. Mooney. Combining neural and symbolic learning to revise probabilistic rule bases. In *Advances in Neural Information Processing Systems, Vol. 5*, San Mateo, CA, in press. Morgan Kaufman.

[Matwin and Plante, 1991] S. Matwin and B. Plante. A deductive-inductive method for theory revision. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 160–174, Harper's Ferry, W.Va., Nov. 1991.

[Murphy and Pazzani, 1991] P. M. Murphy and M. J. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discrimin ators in decision trees. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 183–187, Evanston, IL, June 1991.

[Ourston and Mooney, 1990] D. Ourston and R. Mooney. Changing the rules: a comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 815–820, Detroit, MI, July 1990.

[Ourston and Mooney, in press] D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, in press.

[Ourston, 1991] D. Ourston. *Using Explanation-Based and Empirical Methods in Theory Revision.* PhD thesis, Department of Computer Sciences, University of Texas, Austin, TX, August 1991.

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

[Richards and Mooney, 1991] B. Richards and R. Mooney. First-order theory revision. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 447–451, Evanston, IL, June 1991.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and J. R. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol. I*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[Spackman, 1988] K. A. Spackman. Learning categorical decision criteria in biomedical domains. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 36–46, Ann Arbor, MI, June 1988.

[Towell and Shavlik, 1991] G. Towell and J. Shavlik. Refining symbolic knowledge using neural networks. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 257–272, Harper's Ferry, W.Va., Nov. 1991.

[Towell and Shavlik, 1992] G. Towell and J. Shavlik. Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan Kaufmann, 1992.

[Towell *et al.*, 1990] G. G. Towell, J. W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, July 1990.

[Towell, 1991] G. G. Towell. *Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction.* PhD thesis, University of Wisconsin, Madison, WI, 1991.

[Whitehall *et al.*, 1991] B. L. Whitehall, S. C. Lu, and R. E. Stepp. Theory completion using knowledge-based learning. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 144–159, Harper's Ferry, W.Va., Nov. 1991.