Appears in Informatica, 17, pp. 387-397, 1993

Extending Theory Refinement to M-of-N Rules

Paul T. Baffes and Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, Texas 78712-1188 USA

Keywords: artificial intelligence, multistrategy learning, theory refinement

Edited by: Gheorghe Tecuci
Received: May 26, 1993 **Revised:** October 15, 1993 **Accepted:** October 15, 1993

In recent years, machine learning research has started addressing a problem known as theory refinement. *The goal of a theory refinement learner is to modify an incomplete or incorrect rule base, representing a domain theory, to make it consistent with a set of input training examples. 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 experimental results from two real-world domains.*

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 [4, 11, 18, 3, 21, 7]. 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 demonstrating the advantages of this representation [17, 9]. Other work has focused on revising rules that have real-valued weights [19, 6]. 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 [11, 12] 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 each example, and is therefore much more efficient.

Also, because it was restricted to strict Horn-clause theories, the old EITHER algorithm could not produce as accurate results as a neural-network revision system called KBANN on a domain known as the DNA promoter problem [18, 19]. Essentially, this is because some aspects of the promoter concept fit the M-of-N format. Specifically, 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. The new NEITHER algorithm, however, includes the ability to modify a theory by changing thresholds of M-of-N rules. 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 DNA 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. We also present results showing NEITHER's ability to repair faults in a theory used to teach the diagnosis of shock to novice nursing students. We show that NEITHER is able to restore significantly damaged theories to near perfect accuracy.

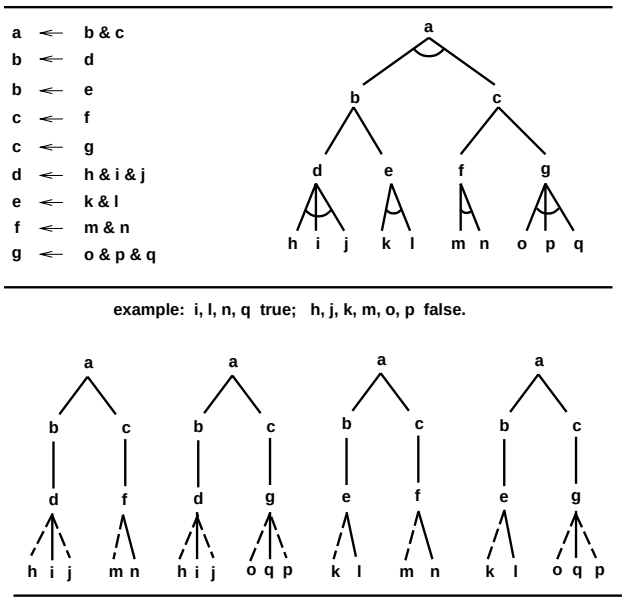

Figure 1: Partial proofs for unprovable positive example. Unprovable antecedents are shown with dotted lines.

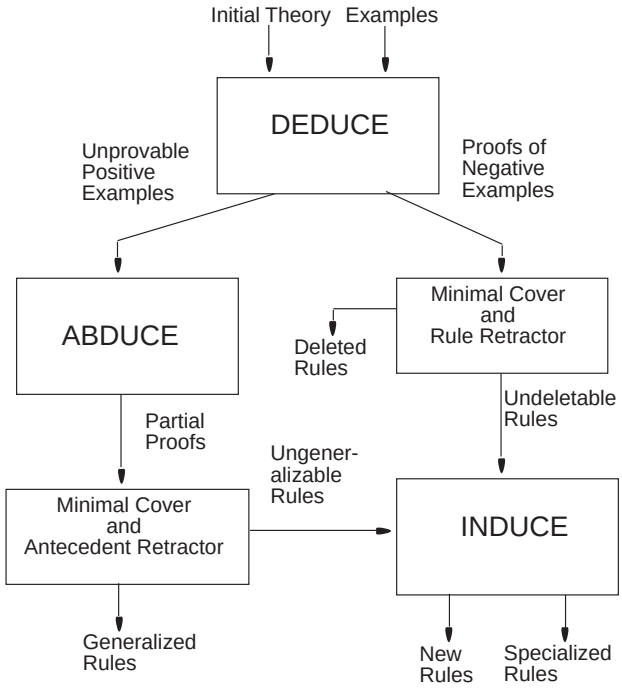

Figure 2: Block diagram of EITHER.

2 Theory Revision Algorithm

2.1 The EITHER Algorithm

The original EITHER theory refinement algorithm has been presented in various levels of detail in [11, 12, 10]. It was designed to revise propositional Horn-clause theories. For EITHER, a *theory* is a set of propositional Horn-clause rules such as those shown in the top half of Figure 1. Each theory is assumed to function as a classification system whereby *examples* are labeled as belonging to one of a given set of categories. Examples are vectors of feature-value pairs listing the value corresponding to each feature, as well as the category into which the example should be classified. As an illustration, one might imagine a diagnostic theory for determining whether or not a given product coming off an assembly line passes inspection. The categories for such a rule base might be "pass" and "fail" and the examples would consist of whatever measurements could be made on the product as part of the inspection test.

In revising an incorrect theory, note that EITHER can fix either overly-general or overly-specific rules through a combination of deductive, abductive and inductive techniques as shown in Figure 2. 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 [10].

The basic algorithm used by EITHER for both generalization and specialization is shown in the top half of Figure 3. There are three 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 1 illustrates this process for theory generalization where EITHER is searching for leaf-rule antecedent deletions to correct failing positive examples. A leaf rule is a rule whose antecedents include an observable or an intermediate concept that is not

[1] EITHER uses a version of ID3 [13] for its induction.

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 3: Comparison of EITHER and NEITHER algorithms.

the consequent of any existing rule. 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. A partial proof is one in which some antecedents cannot be satisfied. From these proofs there are four possible repairs which will fix the example, corresponding to the four partial proofs. In each repair, the dotted lines represent antecedents which cannot be proved and must therefore be removed from the given rule(s). Thus, for the leftmost partial proof, `h` and `j` cannot be proved in the rule `d` ← `h & i & j`, and `m` cannot be proved for rule `f` ← `m & n`, so the repair for this partial proof is: delete `(h,j,m)` from their respective rules. Likewise, the three other repairs are: delete `(h,j,o,p)`, delete `(k,m)` and delete `(k,o,p)`. Theory specialization follows a similar process to return sets of leaf-rule *rule* deletions which fix individual failing negative examples.

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 3. 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 deletion and rule deletion) 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 3).

Rather than computing all partial proofs, NEITHER works bottom-up, constructing a single set of deletions. 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 generalization, deletions 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 deletions 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 deletions is selected since any single failure will disable a rule.

To illustrate how repairs are computed in linear time, refer again to Figure 1. The antecedent deletion calculations for this example would begin at the root of the graph, recursively calling nodes `b` and `c`. Deletion for node `b` would then recurse on nodes `d` and `e`. Since `h`, `j` and `k` are false, node `d` returns `(h,j)` and node `e` returns `(k)`. When the recursion returns back to node `b` a choice must be made between the results from nodes `d` and `e` because the theory is being generalized and node `b` is an OR node. Since node `e` requires fewer deletions, its deletions are chosen as the return value for node `b`. Recursion for node `c` follows a similar pattern: node `f` returns `(m)`, node `g` returns `(o,p)` and node `c` chooses the smaller results from node `f` as its return value. Finally, nodes `b` and `c` return their values to node `a`. Now, since node `a` is an AND node and the theory is being generalized, the results from `b` and `c` are combined. The final repair returned from node `a` is delete `(k,m)`. Thus the rule `e` ← `k & l` is generalized to `e` ← `l`, and the rule `f` ← `m & n` is generalized to `f` ← `n`.

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 deletions 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 with the given set of rules. Of course, once a repair is applied to the theory it will effect subsequent repair calculations because the theory will change. Thus although each repair is minimum with respect to the state of the theory from which is was calculated, the total sum of all repairs may not be minimum due to ordering effects. The only way to reach such a global minimum is to do an exhaustive search which is exponential in the size of the theory. This new algorithm trades the complete information available in the partial proofs for speed in computation.

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

Table 1: Comparison of Revisions.

2.3 Adding M-of-N Rules to NEITHER

Expanding NEITHER to handle M-of-N rules involves both a change in the syntax and interpretation of rules as well as a modification to the types of revisions which can be made to a given theory. With M-of-N rules, there are six types of revisions. 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 delete enough antecedents to make the rule provable; there is no need to delete all false antecedents if the rule has a threshold. For example, if the rule for e in Figure 1 had a threshold of 1 there would be no need to delete 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 deleted must be decreased by the number of antecedents deleted 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. This is because during generalization, the most aggressive changes are those which generalize the most, but during specialization, the most aggressive changes are those which generalize the least.

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.

3 Experimental Results

In the experiments which follow, there are two versions of the NEITHER algorithm. The first has only the speedup changes and is termed simply NEITHER. The second includes M-of-N refinements and is termed NEITHER-MOFN.

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.

3.1 The DNA Promoter Domain

3.1.1 Experimental Design

We tested both NEITHER and NEITHER-MOFN against other classification algorithms using the DNA promoter sequences data set [20]. This data set involves 57 features, 106 examples, and 2 categories. The theory provided with the data set is supposed to recognize *promoters* in strings of nucleotides. A promoter is a genetic region which initiates the first step in the expression of an adjacent gene *transcription*. However, the original theory has an initial classification accuracy of only 50%. We selected this particular data set because the original EITHER algorithm was outperformed by other systems on this data due to the M-of-N qualities required to reason correctly in this domain. In addition to testing EITHER, NEITHER and NEITHER-MOFN, we ran experiments using ID3 [13], backpropagation [16] and RAPTURE [6] (a revision system based on certainty factors). We also included data on the performance of KBANN on this data set as reported in [20].

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).

3.1.2 Results

The results of our experiments are shown in the three graphs of Figures 4, 5 and 6. Figure 4 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.[2] 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 4 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.

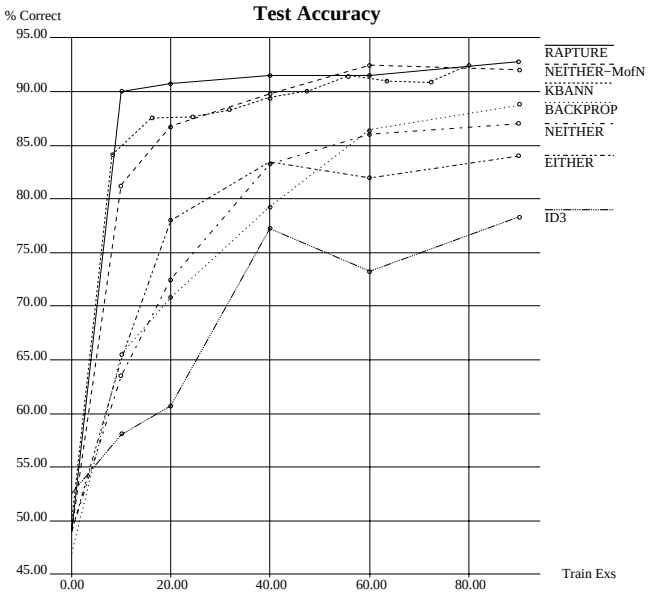

Figure 4: DNA Test Set Accuracy.

Figure 5 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 6. 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.

3.1.3 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

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

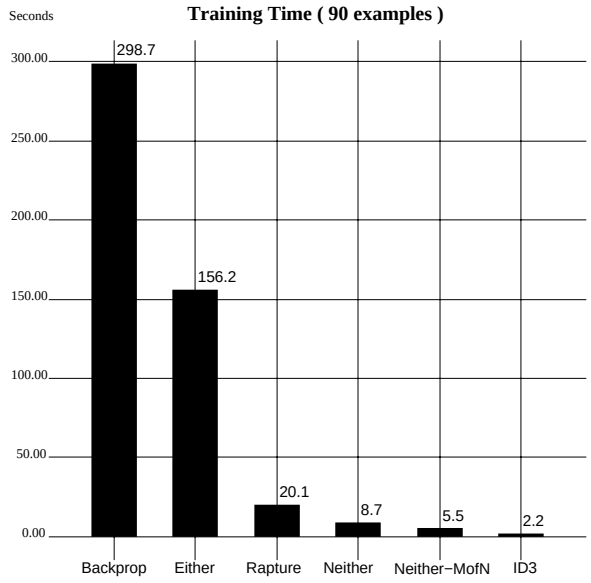

Figure 5: DNA Training Time Comparison.

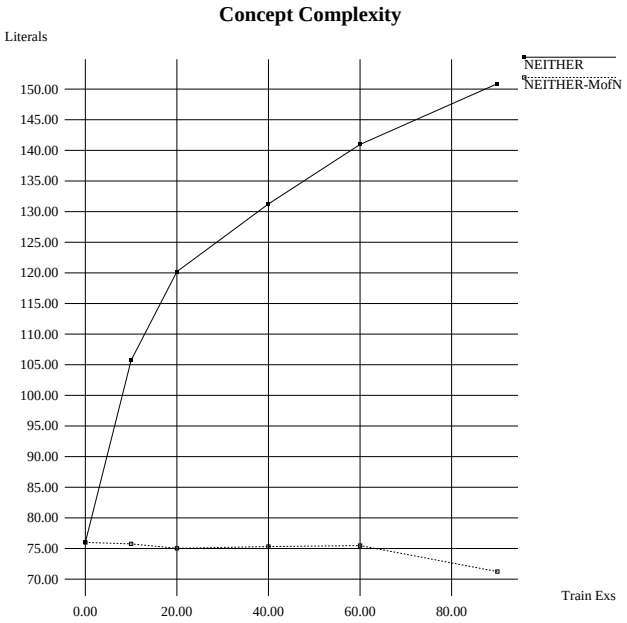

Figure 6: DNA Concept Complexity.

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 [20] would indicate that KBANN's accuracy in the promoter domain is about the same as NEITHER-MofN's.

The most surprising result of the experiments was the difference in accuracy between the original EITHER algorithm and NEITHER. As stated above, EITHER was more accurate with fewer training examples, but its accuracy dropped off relative to NEITHER as the number of examples increased. One possible explanation for this behavior lies in the difference between how the two systems compare potential revisions (see Figure 3). Recall that EITHER computes multiple repairs for each example, but does so only once. NEITHER, by contrast, computes one repair per example each time through its main loop. As a result, with fewer training examples, EITHER has more potential revisions to examine, apparently giving it an edge over NEITHER. Even though NEITHER computes new repairs each time it iterates, there may not be enough iterations to generate as rich a set of deletions as is done in one step by EITHER. On the other hand, as the number of training examples grows, NEITHER undergoes many more iterations, each computing new repairs *in light of any previous revisions*. By contrast, EITHER computes its repairs for each example independently, missing out on any interactions which might occur when the revisions are applied to the theory in a particular order. Capturing these interactions may be one reason NEITHER out-performs EITHER with large numbers of examples.

3.2 The Shock Diagnosis Domain

A second set of experiments was run to test NEITHER's ability to repair faulty theories. The data for this experiment was borrowed from a separate research project designed to test nursing students retention of concepts for determining if a patient is suffering from shock [8]. Each patient can be labeled in one of four ways: as suffering from hypovolemic, cardiogenic, or vascular tone shock, or as not in shock. A theory for diagnosing shock was written using the definitions and examples presented to the students and consultations with a medical expert. The final theory is shown in Figure 7.

This data set was chosen for two reasons. First, it represents another real-world domain which has an M-of-N flavor (the "shock" concept in the theory is represented using a threshold rule). Second, NEITHER's ability to refine theories in this domain is the centerpiece of another of our research efforts in *student modeling* [1]. In short, a student modeling algorithm must be able to recover from a variety of deviations from the correct theory in order to be useful to a variety of students.

hypovolemic ← shock disrupted-blood-volume
cardiogenic ← shock ineffective-pumping-action
vascular-tone ← shock disrupted-vascular-tone
shock ← 3 of ((pulse rising) (respiration rising)
(blood-pressure falling) (urine-output low)
(mental-status strained) skin-abnormal)
shock ← (patient pregnant) (symptoms blood-loss)
(mental-status strained) skin-abnormal
skin-abnormal ← (skin cool-clamy)
skin-abnormal ← (skin hot-flushed)
disrupted-blood-volume ← (symptoms fluid-loss)
disrupted-blood-volume ← (symptoms blood-loss)
ineffective-pumping-action ← (symptoms cardiac)
disrupted-vascular-tone ← (symptoms infection)
disrupted-vascular-tone ← (symptoms allergy)
disrupted-vascular-tone ← (symptoms neural)

Figure 7: Shock Domain Theory.

3.2.1 Experimental Design

The basic design of this experiment was to introduce faults into the correct shock theory and test how well NEITHER-MOFN could refine the result. The following five alterations were injected into the theory with equal probability: antecedent deletions, antecedent additions, rule deletions, rule additions, and threshold changes. A modification factor was passed to the algorithm which made the alterations indicating the percentage of antecedents in the theory to be changed. Consequently, a modification factor of 0.1 indicated that roughly 10% of the antecedents in the theory would be modified.

Data for training and testing was drawn from a pool of 150 examples equally representative of the three categories of shock. These 150 examples were randomly generated using the correct theory. For each category, 40 positive examples and 10 near-miss negative examples were created. Thus, of the 150 total cases, 120 examples belonged to one of the three categories of shock and 30 were non-shock examples.

A three-phased experiment was run. In each experiment, the 150 examples were first split randomly into 100 training examples and 50 test examples. The original theory was then subjected to three independent modifications of 0.1, 0.2 and 0.3. Each resulting theory was refined by NEITHER-MOFN using the same 100 training examples. The theories were tested both before and after refinement using the same 50 test examples. This entire process was repeated 10 times, and the results averaged. For comparison purposes, we also ran the same training data through a propositional version of the FOIL inductive learner [14] and tested the results using the same test data, averaging the results of the 10 trials.

3.2.2 Results

Table 3 shows the results of the recovery experiments. Note that the results for FOIL are identical for each experiment since induction does not make use of an input theory. In each of the three experiments, NEITHER-MOFN was able to reconstruct a theory to perfect or near perfect accuracy on the test data. NEITHER-MOFN was also able to create more accurate theories than induction alone. Tests were also run using the non-threshold version of NEITHER but the results were nearly identical to those reported for NEITHER-MOFN (there was no statistically significant difference between NEITHER and NEITHER-MOFN on this data).

3.2.3 Discussion

The results of Table 3 are largely what one would expect for a good theory refinement algorithm. As the theory deviated more and more from the original, the performance of the altered theory on the test data continued to decline. Likewise, the ability of NEITHER-MOFN to repair the theory declined with increasingly altered theories, though only slightly. Yet in all cases, NEITHER-MOFN was able to refine a wide variety of damaged theories to a high level of performance on novel test data. Additionally, NEITHER-MOFN was able to create more accurate theories than induction alone by taking advantage of input theories which were at least partly correct. Finally, though NEITHER-MOFN was unable to exactly duplicate the original theory in all cases, the refinements made seemed reasonable in light of the alterations made in the modified theories.

4 Related Work

Several researchers have developed methods for inducing M-of-N concepts from scratch. CRLS [17] learns M-of-N rules and out-performed standard rule induction in several medical domains. ID-2-of-3 [9] incorporates M-of-N tests in decision-tree learning and outperformed standard decision-tree induction in a number of domains. Both projects clearly demonstrate the advantages of M-of-N rules.

SEEK2 [5] 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 [19] revises a theory by translating it into a neural network, using backpropagation to refine the

	0.1 modified theory	*0.2 modified theory*	*0.3 modified theory*
before refinement	68.25	50.25	43.5
after refinement	100.0	94.0	94.0
induction	80.4	80.4	80.4

Table 3: Shock Test Set Accuracy.

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 [20], 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 [6] uses a combination of symbolic and neural-network learning methods to revise a certainty-factor rule base [2]. 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 the original EITHER algorithm, such as constructive induction, modification of higher-level rules, and the ability to handle numerical features and noisy data. Also, we could extend our methods to handle negation as failure and incorporate the ability to handle M-of-N rules into first-order theory revision [15]. The inductive component of NEITHER should be modified to produce threshold rules directly, rather than symbolic rules. 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 [11]; 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.

Acknowledgments

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. We would like to thank Jude Shavlik, Geoff Towell, and Marylin Murphy for generously providing the DNA and Shock data sets. Special thanks also to Chris Whatley for his help implementing NEITHER and to Dr. Thomas Baffes for his invaluable help in developing the rules for the shock domain.

References

[1] P. Baffes and R. J. Mooney. Using theory revision to model students and acquire stereotypical errors. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 617–622, Bloomington, IN, 1992.

[2] 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.

[3] 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.

[4] 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.

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

[6] J. J. Mahoney and R. J. Mooney. Combining neural and symbolic learning to revise probabilistic rule bases. In S.J. Hanson, J.C. Cowan, and

C.L. Giles, editors, *Advances in Neural Information Processing Systems, Vol. 5*, pages 107–114, San Mateo, CA, 1993. Morgan Kaufman.

[7] 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.

[8] Marilyn A. Murphy and Gayle V. Davidson. Computer-based adaptive instruction: Effects of learner control on concept learning. *Journal of Computer-Based Instruction*, 18(2):51–56, 1991.

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

[10] D. Ourston. *Using Explanation-Based and Empirical Methods in Theory Revision.* PhD thesis, University of Texas, Austin, TX, August 1991. Also appears as Artificial Intelligence Laboratory Technical Report AI 91-164.

[11] 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.

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

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

[14] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.