When Prior Knowledge Hinders Learning

AAAI Workshop on Constraining learning with Prior Knowledge . San Jose, CA.

Michael Pazzani
Department of Information and Computer Science
University of California
Irvine, CA 92717
pazzani@ics.uci.edu

1.0 Introduction

There are two general approaches to learning classification rules. Empirical learning programs operate by finding regularities among a group of training examples. Analytic learning systems use prior knowledge to explain the classification of examples and form a general description of the class of examples with the same explanation. Many systems (e.g., Bergadano & Giordana, 1988; Cohen, in press-a; Flann & Dietterich, 1989; Hirsh, 1989; Lebowitz, 1986; Michalski & Ko, 1988; Ourston & Mooney, 1990; Pazzani, 1990; Shavlik & Towell, 1989; Wilkins & Tan, 1989) have combined empirical and analytic learning methods. In such systems, the prior knowledge is used to bias the learner to prefer concepts that are consistent with the prior knowledge.

This bias is intended to make a learner create concept descriptions that are more accurate classifiers than both the original prior knowledge (that serves as input to the analytic learning component) and the rules that would arise if only the empirical learning component were used. In previous work (Pazzani, Brunk, & Silverstein, 1991; Pazzani & Kibler, in press), we have described FOCL, a system that extends Quinlan's (1990) FOIL program in a number of ways, most significantly by adding a compatible explanation-based learning (EBL) component.

Here, we describe a simple example deliberately constructed to show that prior knowledge can also hinder learning in FOCL as well as other combined learned systems. We analyze why this example presents problems for these systems and suggest an approach that allows FOCL to exploit this type of prior knowledge to facilitate learning.

1.1 FOIL

FOIL learns classification rules by constructing a set of Horn Clauses in terms of a set of known operational predicates. Each clause body consists of a conjunction of literals that cover some positive and no negative examples. FOIL starts to learn a clause body by finding the literal with the maximum information gain[1], and continues to add literals to the clause body until the clause does not cover any negative examples. After learning each clause, FOIL removes the positive examples covered by that clause from further consideration. The learning process terminates when all positive examples have been covered by a clause.

1.2 FOCL

FOCL extends FOIL by including a compatible EBL component. This allows FOCL to take advantage of a given domain theory. When constructing a clause body, there are two ways that FOCL can add literals. First, it can create literals via the same empirical method used by FOIL. Second, it can create literals by operationalizing a target concept, i.e., a non-operational definition of the concept to be learned (Mitchell, Keller, & Kedar-Cabelli, 1986). FOCL uses the information-based evaluation function to determine whether to add a literal learned empirically or a conjunction of literals learned analytically. A clause learned by

1. The information gain metric used is:
Gain(Literal) = $T^{++} * \log_2(p_1/p_1+n_1)-\log_2(p_0/p_0+n_0)$
where p_0 and n_0 are the current number of positive and negative examples, p_1 and n_1 are the number of positive and negative examples that would remain after adding the literal, and T^{++} is the number of current positive examples that have at least one corresponding example in the positive examples after adding the literal (Quinlan, 1990).

Table 1. The operationalization process in FOCL.

```
operationalize(Literal, Pos, Neg):
    Initialize Conjuncts to the empty set
    For each Clause in the definition of Literal
        compute_gain(Clause, Pos, Neg)
    Let Clause be the clause with the maximum gain
    For each literal L in Clause
        if L is operational
            then Set Pos to the extensions of Pos satisfied by L
                 Set Neg to the extensions of Neg satisfied by L
                 Add L to Conjuncts
            else add operationalize(L, Pos, Neg) to Conjuncts
    Return Conjuncts
```

FOCL may have the following form:

$$r \leftarrow O_i \ O_d \ O_f$$

where O_i is an initial conjunction of operational literals learned empirically, O_d is a conjunction of literals found by operationalizing the domain theory, and O_f is a final conjunction of literals learned empirically. The integration between the empirical and analytic learning methods in FOCL is fairly tight. For example, the literals in O_f may refer to variables that are introduced by O_d.

Operationalization in FOCL differs from that of most EBL programs in that it uses a set of positive and negative examples, rather than a single positive example. A non-operational literal is operationalized by producing a specialization of a domain theory that is a conjunction of operational literals. When there are several ways of operationalizing a literal (i.e., there are multiple, disjunctive clauses), the information gain metric is used to determine which clause should be used by computing the number of examples covered by each clause. Table 1 presents an overview of this operationalization process. In Pazzani, Brunk, and Silverstein (1991), an extension to the operationalization process is described that consists of a greedy deletion of literals in O_d guided by the information-based evaluation function.

Pazzani, Brunk, and Silverstein (1991) demonstrate that FOCL can take advantage of a wide variety of incomplete and incorrect domain theories that were formed by syntactic mutations of correct domain theories. In these domain theories, as well as several realistic domain theories on problems such as foreign trade negotiation, and educational loans, FOCL with prior knowledge was more accurate than FOIL (i.e., FOCL without prior knowledge). In the next section, we show an example domain theory in which the opposite is true: FOCL with prior knowledge is less accurate than FOCL without prior knowledge.

2.0 Incorrect domain theories that hinder learning.

The accuracy of the prior knowledge is not the only important characteristic for predicting FOCL's ability to accurately learn a concept. In order for FOCL to tolerate incomplete and incorrect domain theories, the empirical learning process must be able to "patch" the operationalized conjunctions of literals derived from the domain theory. This can be accomplished in several ways:[2]

1. Some clauses learned by FOCL may result from deleting one or more literals from an operationalization of the domain theory. This occurs because FOCL deletes literals from an operationalization if such a deletion increases information gain (e.g., by allowing the clause to cover more positive examples).

2. Some clauses learned by FOCL may consist of one or more literals, found by empirical means, conjoined with an operationalization. This occurs because FOCL uses empirical methods to specialize

2. Note that FOCL does not contain separate operators to deal with each type of correction. Rather, this behavior falls out of a general approach to find literals with the maximum information gain by either empirical or analytical methods.

an operationalization if the operationalization covers negatives examples.

3. Some clauses learned by FOCL may consist of a conjunction of literals learned empirically. This occurs when the prior knowledge has no information gain (e.g., it incorrectly explains many negative examples and/or fails to explain many positive examples).

Note that the operationalization used in the second item may have several literals deleted before it is specialized empirically. Given this behavior (and a year of experience with FOCL as well as discussions with William Cohen), it is easy to see how one could construct a problem on which prior knowledge could hinder learning. The idea behind this construction is to force FOCL with prior knowledge to learn a concept with several clauses, while FOCL without prior knowledge could learn a concept with fewer clauses. This can be accomplished if:

a. An operationalization of the prior knowledge will be sufficiently accurate to have some information gain, yet still cover some negative examples.
b. No deletion from the operationalization will have more information gain.
c. The best way to specialize the operationalization is to add the exact same literals that would be formed empirically as the correct answer.

As a result, one clause will initially be formed by a combination of empirical and analytic methods methods that is more specialized than one that could be learned by empirical methods alone. Next, FOCL will remove the examples covered by this clause, and learn another clause by empirical methods alone. The most accurate clause will consist of the exact same literals found empirically to specialize the operationalization. If all goes well (e.g., if there are many training instances), the final clauses learned by FOCL with domain knowledge would be $r \leftarrow O_d\ O_f$ and $r \leftarrow O_f$ while FOCL without domain knowledge would learn $r \leftarrow O_f$. However, note that to be as accurate as FOCL without domain knowledge, FOCL with domain knowledge must find the exact same conditions twice on two different subsets of the examples. In contrast, without domain knowledge, FOCL will have to learn these same conditions once from a larger set of examples. In this case, the prior knowledge serves to partition the examples into two sets, and cause a problem that is analogous to the replicated subtree problem in decision trees (Pagallo & Haussler, 1990). Because it is less likely that an empirical method will find an accurate rule twice on two partitions of a data set than once on the combined data set, FOCL with domain knowledge would be expected to be less accurate than FOCL without domain knowledge.

An example of such a domain theory will help to illustrate this problem. Consider the problem of learning a definition for `odd_product(X,Y)` which is true when the product of `X` and `Y` is odd. The variables `X` and `Y` can take on the values of integers from 1 to 9.

The following predicates, defined extensionally, may be used to define `odd_product`:

- `odd(X)`- True if X is odd.
- `small(X)`- True if X is less than or equal to 5.
- `prime(X)`- True if X is a prime number.
- `square(X)`- True if X is a perfect square.
- `cube(X)`- True if X is a perfect cube.

The correct definition for this concept is `odd_product(X,Y) :- odd(X) & odd(Y).` However, the definition provided to FOCL as prior knowledge is:

```
odd_product(X,Y) :- small(X) & small(Y).
```

This approximate definition correctly classifies 75.3% of the possible examples (61 of 81). Note that deleting either `small(X)` or `small(Y)` is unlikely to have information gain, since deleting each of these literals results in a condition that classifies only 50.6% of the examples correctly. As a consequence, from most randomly drawn subsets of these 81 examples, FOCL will operationalize this incorrect definition, but not delete either literal. Next, the empirical algorithm will specialize this definition

(since it incorrectly classifies examples with small, even integers). The specialization that covers the most examples on the entire data set is:

```
odd_product(X,Y) :-   small(X) & small(Y) &
                      odd(X) & odd(Y).
```

However, on some training sets, other specializations may perform equally well on subsets of the data, including:

```
odd_product(X,Y) :-   small(X) & small(Y) &
                      odd(X).
odd_product(X,Y) :-   small(X) & small(Y) &
                      prime(X).
odd_product(X,Y) :-   small(X) & small(Y) &
                      not(square(X)).
```

Once any of the above clauses are learned,[3] FOCL would have to learn a second clause to cover any remaining positive examples (e.g., products of large odd numbers). The domain theory is unlikely to have information gain at this point, because most of the positive examples explained by it would be covered by the correct clause. As a consequence, the empirical learner would learn the next clause. Although the correct clause is `odd_product(X,Y) :- odd(X) & odd(Y)`, on small samples, other clauses may perform equally well such as

```
odd_product(X,Y) :- prime(X) & prime(Y).
```

We ran 32 trials of FOCL with and without this domain theory on training sets ranging from 10 to 50 examples. The accuracy at various points was measured by testing on 25 test examples (that were not part of the training set). Figure 1 shows the mean accuracy of the two algorithms as a function of the training examples. The rules learned by FOCL without a domain theory are more accurate than those learned by FOCL with a domain theory. An analysis of variance indicated that the use of prior knowledge has a significant effect on the accuracy.

To determine whether this is a problem specific to FOCL, or whether the problem is more general, the same domain theory encoded in propositional logic was tested on KBANN (Shavlik & Towell, 1989; Towell, 1991) by Geoffrey Towell and on EITHER (Ourston & Mooney) by Paul Baffes of the University of Texas. For EITHER and KBANN, each example is represented as a set of 10 Boolean attributes corresponding to whether or not the first or second number is odd, small, prime etc. The mean accuracy of KBANN and backpropagation (Rumelhart, Hinton & Williams, 1986), the empirical component of KBANN, is also shown is Figure 1. The BackProp network used here has no hidden units, since the KBANN network also has no hidden units. However, similar results were obtained using networks with between one and nine hidden units.

KBANN operates by converting a domain theory into an artificial neural network whose structure and weights encode the domain knowledge. The weights of this network are then adjusted by the backpropagation learning algorithm (Rumelhart, Hinton & Williams, 1986) running on the training data. The trained network is then tested on the training data. A recent extension to KBANN called MofN (Towell & Shavlik, 1991) then extracts symbolic rules from this network. On this problem, MofN rules had approximately the same accuracy as the network from which they were derived. These rules have problems somewhat similar to the rules learned by FOCL. For example, on small data sets, rules such as

```
odd_product :- 2 of{odd-x, odd-y, prime-x}.
```

were observed. On larger data sets, rules such as

```
odd_product :- odd-x &  odd-y &
```

3. FOCL contains a postprocessor that runs after each clause is learned (Quinlan, 1990). This postprocessor deletes superfluous literals, so that if `odd_product(X,Y) :- small(X) & small(Y) & odd(X) & odd(Y).` were learned, it would be simplified to `odd_product(X,Y) :- odd(X) & odd(Y).` Although this simplifier mitigates the problem, it does not eliminate it. In particular, the "incorrect" (from our viewpoint) specializations cannot not be simplified since each condition may be necessary to rule out some negative examples. All experiments in this paper make use of the clause simplification postprocessor.

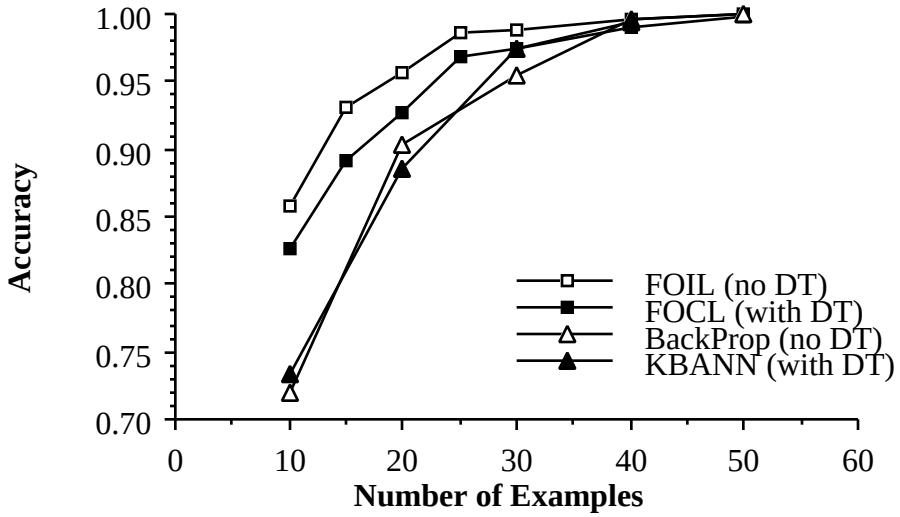

Figure 1. A comparison of FOCL and KBANN with and without a domain theory and EITHER with and without a domain theory.

```
not(2 of{square-x, cube-y, prime-y}.
```

were observed.

On this problem, KBANN networks do not perform significantly differently than BackProp networks. This result differs from that found in EITHER or FOCL since the knowledge-based learner is not less accurate than its inductive learning component. However, on this problem BackProp networks are substantially less accurate than ID3 or FOIL. Unfortunately, due to the complexity of BackProp network, we have no explanation for this result.

The mean accuracy of EITHER and ID3 (Quinlan, 1986), the empirical component of EITHER, are shown is Figure 2. As with FOCL, the rules learned by the empirical component were more accurate than those learned by the combined method. Although there are many differences between EITHER and FOCL[4], the rules learned by EITHER and FOCL with incorrect domain theories were similar. For example, one rule learned by EITHER is:

```
odd_product :- small-x & small-y & odd-x.
odd_product :- square-x & odd-x.
```

It appears that the rules learned by EITHER suffer from the effects of repeated induction on partitions of the training data in a manner similar to FOCL.

One similarity between EITHER and FOCL provides additional insight into the problem. Both systems make a decision to generalize by deleting a literal (from the domain theory in EITHER's case or an operational in FOCL's case) that is independent of the decision to specialize by add a literal. In this particular domain theory, deleting a literal reduces accuracy (or the information gain) and both learners avoid this operation and produce an overly specialized clause.

4. For example, EITHER uses propositional logic, while FOCL uses Horn clauses; EITHER uses a divide-and-conquer empirical strategy, while FOCL uses a separate-and-conquer strategy; EITHER has an abduction component for deleting literals, while FOCL uses greedy deletion of literals from operationalizations; they use a different evaluation function, EITHER revises a domain theory while FOCL learns an operational definition of the target concept, etc.

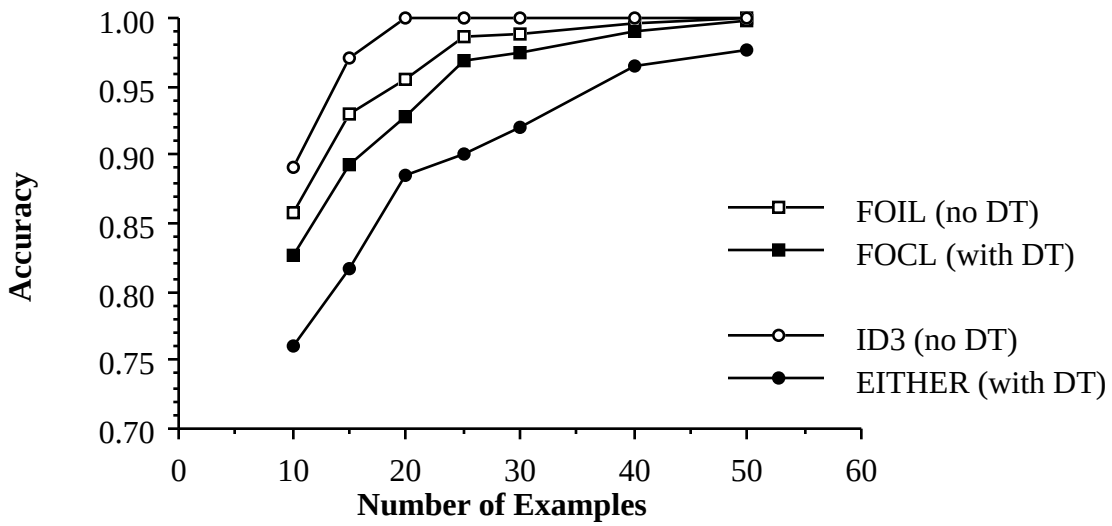

Figure 2. A comparison of FOCL with and without a domain theory and EITHER with and without a domain theory. In both cases, the integrated system is less accurate than its empirical component.

3.0 Selectively Replacing Literals

An analysis of some of the conditions under which a domain theory may hinder learning also suggests a solution. As a first pass, if a learner had the option of replacing one literal with another in a single step[5], rather than two steps (deleting one and adding another), then on most training sets there would be a benefit in replacing `small(X) & small(Y)` with `odd(X) & small(Y)`. In the next step, the learner would be able to replace `odd(X) & small(Y)` with `odd(X) & odd(Y)`.

However, rather than considering all replacements, it is possible to be more selective and only deal with those situations in which hill climbing of deleting one literal and then adding another literal is likely to fail. In particular, the reason that deleting an incorrect literal does not improve the accuracy or increase the information gain in this simple example is that the incorrect literal is correlated with the correct literal. On account of this correlation, the incorrect literal performs at better than chance levels in the clause under construction. Therefore to perform selective replacement of literals, we only need to consider replacement of literals with similar literals. We will return to the definition of "similar" shortly, but first we give an overview of FOCL-R, an extension to FOCL that allows for the replacement of literals in an operationalization.

The outer loop of FOCL-R is identical to FOIL. It repeats the process of learning clauses until every positive example is covered by some clause. An inner loop learns clauses by adding literals (or conjunctions of literals) to the clause under construction until no negative example is covered. There are two methods for adding literals:

1. Empirical- All ways of creating new literals with extensionally defined predicates are

5. Note that alternate solutions would be to replace the hill-climbing search used by EITHER and FOCL with another search method, such as simulated annealing, or to allow a look-ahead greater than one operator application. The replacement operator we are proposing is equivalent to a limited form of look-ahead for exactly the situation that we have found to cause problems with FOCL and EITHER.

found and the literal with the maximum information gain is returned. This is essentially the FOIL algorithm

2. Analytic- The operationalization with the maximum information gain is found. A greedy process computes the information gain of all single deletions or single selective replacements from the operationalization. If any modification of the operationalization has more information gain than original operationalization, this greedy process is repeated on the modification with the maximum gain. The process terminates when no modification improves gain, and a conjunction of literals with the maximum information gain is returned.

The procedure that FOCL-R uses to decide whether to add a literal found empirically, or a conjunction of literals found analytically is simple. Whichever choice has more information gain is chosen. Similarly, the choice of whether or not to delete or replace literals from an operationalization is also determined by this same metric.

There are a variety of similarity metrics between literals that could be defined. For now, we have restricted our attention to literals with the same arguments, but with different extensionally defined predicates and have defined a similarity metric between predicates.

$$Sim(P_1,P_2) = \frac{Agree(P_1,P_2)}{Size(P_1,P_2)}$$

where $Agree(P_1,P_2)$ counts the number of tuples that two predicates have in common and $Size(P_1,P_2)$ is the number of tuples in either predicate. When this metric is above a certain threshold, two predicates are considered similar. We use 0.5 for this threshold. The similarity between predicates is computed only once, prior to learning. FOCL-R only considers replacing one literal in an operationalization with a literal formed by replacing the predicate of the literal with other similar predicates.

Figure 3 repeats the data from FOCL and FOIL for Figure 1 and includes the data for FOCL-R under the same conditions. This shows that FOCL-R is able to take advantage of a domain theory that hinders learning in FOCL.

4.0 Conclusion and Discussion

There are a variety of issues that deserve additional exploration in this area. First, we would like to explore better similarity metrics between predicates, particularly those that allow predicates of different arity. For example, the predicates `adult(X)` and `parent(X,Y)` are in some sense correlated. In addition, it may be necessary to be somewhat more conservative, and only allow replacements that produce a statistically significant increase in an evaluation function. This may be necessary on some problems so that replacement does not decrease accuracy of correct portions of a domain theory.

In this paper, we have adopted a methodology that is common in linguistics and some areas of AI, such as automated reasoning. We have created an extremely simple test example, and shown how some existing systems do not achieve the goal of taking advantage of prior knowledge on this example. It remains to be seen how common this problem is on "real" domain theories. However, we feel it is important to identify conditions under which prior knowledge hinders learning and we have identified one such case. This occurs when the incorrect theory uses predicates that do not appear in a compact correct theory and these predicates are correlated with those that do appear in a compact correct theory.

We have run FOCL-R on one such real problem: determining whether a trading partner is likely to make a concession in a foreign trade negotiation. Although several replacements occurred, there was no difference between the accuracy of FOCL and FOCL-R on this domain. However, the substitutions made appear plausible. For example, one rule was changed from using a condition corresponding to "The partner has threatened to retaliate" to "The commodity is covered by the general agreement on tariffs and trade."

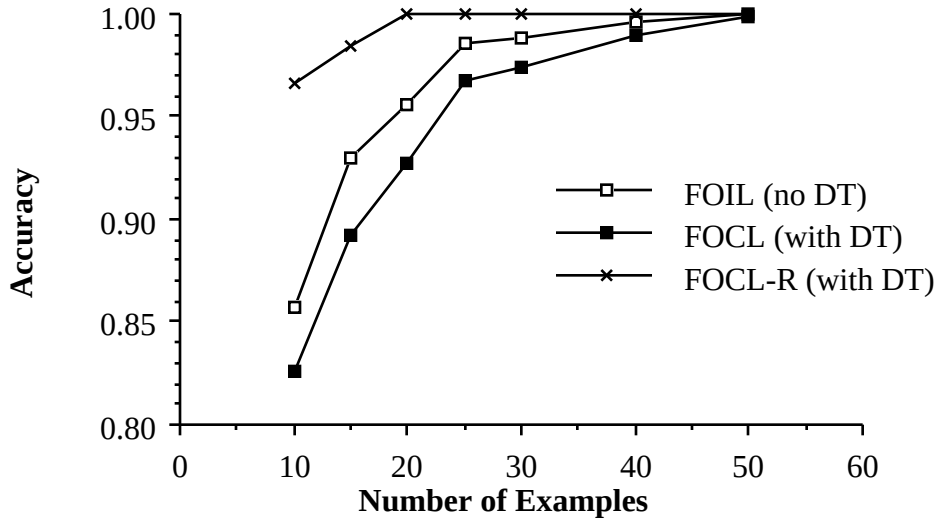

Figure 3. A comparison of FOCL-R with FOCL and FOIL.

We have proposed a simple change to FOCL, that allows replacement of literals in a single step rather than requiring two steps (a deletion and an addition) and shown that this replacement operation results in improved performance. This is a simple instance of a general principle for learning systems that take advantage of prior knowledge. By having an understanding of the types of problems that are likely to be present in prior knowledge, one can design operators to anticipate and correct these problems (Cohen, in press-b). In this particular case, we have assumed that two predicates that are correlated may be confused by a domain expert, and created a strategy to correct for this error.

FOCL as previously described was able to deal with a variety of errors, including clauses containing extra literals, clauses with literals omitted, and clauses that are missing. Given our own experience writing domain theories, we felt that these errors are common. However, FOCL could not take advantage of domain knowledge if this domain knowledge was incorrect because the domain theory contained predicates that are similar to the correct predicates. When developing domain theories, we occasionally use predicates such as `less_than`, while intending to use `less_than_or_equal_to`. In this paper, we showed that such mistakes hurt FOCL, and proposed an extension to FOCL to address this problem. We close by indicating that research is needed into identifying common mistakes that are made by knowledge engineers in encoding domain knowledge.

Acknowledgements

We thank Paul Baffes and Ray Mooney for running either on the problem in Section 2, and Geoffrey Towell for running kbann of this same problem, William Cohen for discussions on incorrect domain theories, Caroline Ehrlich and Jim Wogulis for commenting on an earlier draft of this paper.

References

Bergadano, F., & Giordana, A. (1988). A knowledge intensive approach to concept induction. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 305–317). Ann Arbor, MI: Morgan Kaufmann.

Cohen, W. (in press-a). Abductive explanation-based learning: A solution to the multiple inconsistent explanation problem. *Machine Learning.*

Cohen, W. (in press-b). Compiling prior knowledge into an explicit bias. *Machine Learning Conference.*

Flann, N., & Dietterich, T. (1989). A study of explanation-based methods for inductive learning. *Machine Learning, 4*, 187–226.

Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 29–33). Ithaca, NY: Morgan Kaufmann.

Lebowitz, M. (1986). Integrated learning: Controlling explanation. *Cognitive Science*, *10*, 219–240.

Michalski, R., & Ko, H. (1988). On the nature of explanation or why did the wine bottle shatter? *Proceedings of the AAAI Symposium on Explanation-based Learning* (pp. 12–16). Stanford, CA: Morgan Kaufmann.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning, 1,* 47–80.

Ourston, D., & Mooney, R. (1990). Changing the rules: A comprehensive approach to theory refinement. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 815–820). Boston, MA: Morgan Kaufmann.

Pagallo, G., & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning, 5,* 71–100.

Pazzani, M. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods.* Hillsdale, NJ: Lawrence Erlbaum.

Pazzani, M., Brunk, C., & Silverstein, G. (1991). A knowledge-intensive approach to learning relational concepts. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 432–436). Evanston, IL: Morgan Kaufmann.

Pazzani, M., & Kibler, D. (in press). The role of prior knowledge in inductive learning. *Machine Learning.*

Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning, 5*, 239–266.

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

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations,* (pp 318-362). Cambridge, MA: MIT Press.

Shavlik, J., & Towell, G. (1989). Combining explanation-based learning and artificial neural networks. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 90–93). Ithaca, NY: Morgan Kaufmann.

Towell, G. (1991). *Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction.* Ph.D. Dissertation, Computer Sciences Department, University of Wisconsin.

Towell, G. & Shavlik, J. (1991). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. *Advances in Neural Information Processing Systems (NIPS),* Denver, CO.

Wilkins, D., & Tan, K. (1989). Knowledge base refinement as improving an incorrect, inconsistent and incomplete domain theory. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 332–337). Ithaca, NY: Morgan Kaufmann.