Machine Learning, 19, 79–92 (1995)

Encouraging Experimental Results on Learning CNF

RAYMOND J. MOONEY mooney@cs.utexas.edu
Department of Computer Sciences, University of Texas, Austin, TX 78712

Editor: Dennis Kibler

Abstract. This paper presents results comparing three simple inductive learning systems using different representations for concepts, namely: CNF formulae, DNF formulae, and decision trees. The CNF learner performs surprisingly well. Results on five natural data sets indicates that it frequently trains faster and produces more accurate and simpler concepts.

Keywords: concept induction, experimental comparison, CNF, DNF, decision trees

1 Introduction

Most empirical research on symbolic concept induction has focussed on learning decision trees (Quinlan, 1986; Breiman et al., 1984; Buntine & Niblett, 1992; Fayyad & Irani, 1992) or disjunctive normal form (DNF) expressions (Michalski & Chilausky, 1980; Michalski et al., 1986; Clark & Niblett, 1989; Pagallo & Haussler, 1990). Very little experimental research has been done on learning conjunctive normal form (CNF). This paper presents empirical results comparing simple CNF, DNF, and decision-tree learners on five natural data sets. The decision-tree system is a version of ID3 (Quinlan, 1986), the DNF system is a propositional version of FOIL (Quinlan, 1990), and the CNF system is the logical dual of the FOIL system. In the domains tested, the CNF learner consistently obtains greater or equal classification accuracy and generally runs faster and produces less complex concepts.

The success of the CNF-learning system is somewhat surprising, since CNF seems to have a reputation as an "unnatural" concept representation. Although computational learning theorists frequently examine CNF representations (e.g. kCNF, k-clause CNF) (Valiant, 1984; Pitt & Valiant, 1988), there are no popular, practical algorithms for learning CNF. Actually, CNF is a quite natural representation for "nearly conjunctive" concepts. In addition, our results indicate that the dual of FOIL is an efficient and effective CNF learner.

2 Learning algorithms

2.1 DNF learner

The DNF learner is a propositional version of FOIL (Quinlan, 1990), which we call PFOIL. FOIL is a system for learning first-order Horn clauses; however, the basic algorithm is a heuristic covering algorithm for learning DNF similar to AQ (Michalski, 1975) or GREEDY3 (Pagallo & Haussler, 1990). Pseudocode for PFOIL is shown in Table 1. The primary simplification compared to FOIL, is that it only needs to deal with fixed examples rather than expanding tuples. Each execution of the outer loop adds one term to the final DNF expres-

Table 1. DNF learning algorithm: PFOIL.

```
Let Pos be all the positive examples.
Let DNF be empty.
Until Pos is empty do:
    Let Neg be all the negative examples.
    Set Term to empty and Pos2 to Pos.
    Until Neg is empty do:
       Choose the feature-value pair, L, that maximizes DNF-gain(L, Pos2,Neg2)
        Add L to Term.
        Remove from Neg all examples that do not satisfy L.
        Remove from Pos2 all examples that do not satisfy L.
    Add Term as one term of DNF.
    Remove from Pos all examples that satisfy Term.
Return DNF

Function DNF-gain(L,Pos,Neg)
   Let P be the number of examples in Pos and N the number of examples in Neg
   Let p be the number of examples in Pos that satisfy L.
   Let n be the number of examples in Neg that satisfy L.
   Return p*(log(p/(p+n)) - log(P/(P+N)))
```

sion and removes the positive examples that it covers. Each execution of the inner loop adds one literal to the current term, using an information-gain metric to determine the best literal. The metric computes the total information gained regarding the current positive examples, which is given by the number of them that satisfy the literal multiplied by the information gained regarding each one of them (Quinlan, 1990).

The current system only handles discrete-valued features, in which a "literal" is a specific feature-value pair, such as `color=red`, `pregnant=true`, or `pregnant=false`. No special processing is used to handle missing values. An example that is missing a value for a particular feature is simply assumed *not* to satisfy any specific feature-value pair for that feature. If due to noise, missing values, or local minima, there is no literal with positive information gain, the incomplete term is simply returned. This happened relatively rarely in the current experiments, training set accuracy almost always averaged over 99%. Finally, there is currently no pre-pruning or post-pruning of terms or clauses to avoid overfitting.

When learning from data with multiple disjoint categories, PFOIL learns a separate DNF description for each non-negative category using the examples of that category as positive instances and the examples of all other categories as negative instances. During testing, if an example satisfies the DNF description of more than one category, it is assigned to the matching category with the most training examples. If a test example doesn't match any of the category descriptions, it is assigned to overall most common category, or to the `negative` category, if one exists.

To help ensure that PFOIL was a representative DNF learner, we ran several experiments comparing it to a comparable version of AQ (Michalski, 1975). In almost all cases, there was no statistically significant difference in predictive accuracy; however, AQ generally took significantly longer to run. If AQ's beam-width was set to one (therefore forcing it to hill-climb like PFOIL), its run time was similar to PFOIL's but it's DNF expressions were significantly more complex.

Table 2. CNF learning algorithm: PFOIL-CNF.

```
Let Neg be all the negative examples.
Let CNF be empty.
Until Neg is empty do:
    Let Pos be all the positive examples.
    Set Clause to empty and Neg2 to Neg.
    Until Pos is empty do:
        Choose the feature-value pair, L, that maximizes CNF-gain(L,Pos,Neg2)
        Add L to Clause.
        Remove from Pos all examples that satisfy L.
        Remove from Neg2 all examples that satisfy L.
    Add Clause as one clause of CNF.
    Remove from Neg all examples that do not satisfy Clause.
Return(CNF)

Function CNF-gain(L,Pos,Neg)
    Let P be the number of examples in Pos and N the number of examples in Neg.
    Let p be the number of examples in Pos that do not satisfy L.
    Let n be the number of examples in Neg that do not satisfy L.
    Return n*(log(n/(p+n)) - log(N/(P+N)))
```

2.2 *CNF learner*

There has been a noticeable absence of heuristic algorithms for learning CNF. The standard PAC algorithm for learning kCNF (Valiant, 1984) requires a fixed limit on the size of clauses (k) and its time complexity is exponential in k. Therefore, it is not practical for most realistic problems.

The current experiments employ a logical dual of PFOIL, which we call PFOIL-CNF. Pseudocode for PFOIL-CNF is shown in Table 2.[1] While PFOIL learns terms until all of the positive examples are covered, PFOIL-CNF learns clauses until all of the negative examples are removed. While PFOIL learns terms one literal at a time until all the negatives are removed, PFOIL-CNF learns clauses one literal at a time until all of the positives are covered. PFOIL-CNF's information gain metric is also the dual of PFOIL's. It computes the total information gained regarding the current negative examples, which is given by the number of them that are removed by the current literal multiplied by the information gained regarding each one of them.

All of the comments about PFOIL regarding discrete features, missing values, pruning, and multiple categories apply in corresponding form to the current version of PFOIL-CNF. In every sense, PFOIL and PFOIL-CNF are exact duals.

2.3 *Decision tree learner*

The decision-tree learner is a version of ID3 (Quinlan, 1986). The same version was used in the experiments reported by Shavlik et al. (1991). It learns a single tree for classifying examples into multiple categories and uses the normal information-gain splitting criterion.

Missing feature values are handled according to the methods recommended in (Quinlan, 1989). When evaluating a potential splitting feature, the system adjusts its information gain by distributing examples with unknown values across the possible values according to

their frequency. When considering partitioning the training set along a particular attribute, if a training example has an unknown value for this attribute, a fraction of the example is assigned to each subset based on the frequency of its corresponding value. When classifying a new case with an unknown value for the attribute being tested, all branches are explored and the results are combined to reflect the relative probabilities of the different outcomes.

In order to make it comparable to PFOIL and PFOIL-CNF, no pruning is performed. If the system is eventually left with examples with the exact same features but in different classes, it creates a leaf labelled with the majority class.

3 Data sets

This section briefly describes the data sets used in the experiments. All of them are available through the UCI repository of machine learning databases (Murphy & Aha, 1993).

The *soybean* data set is the original data for soybean disease diagnosis (Michalski & Chilausky, 1980). It contains 630 examples of 15 different diseases described using 35 features in which 5.1% of the feature values are missing. A few features can be viewed as discrete-valued numerical attributes; however, these are simply treated as nominal in the current experiments.

The *congressional voting* data set contains records from the U.S. House of Representatives from the year 1984 (Schlimmer, 1987). It consists of 435 examples from two classes (Democrat and Republican) with data from 15 key votes in which 5.8% of the feature values are missing. Like Buntine and Niblett (1992), we deleted the feature `physician-fee-freeze` in order to make a more interesting problem. Otherwise, the most reliable concept description is a trivial test of this single attribute. PFOIL and PFOIL-CNF were run with Democrats as positive and Republicans as negative. Similar results were obtained when Republicans were considered positive.

The *audiology* data set consists of hearing-disorder cases from the Baylor College of Medicine (Porter et al., 1990) in Quinlan's standardized form available from the UCI repository. There are 226 examples of 24 categories of hearing disorders using 69 features in which 2% of the feature values are missing.

The *promoter* data set consists of 106 DNA sequences, each represented as a string of 57 nucleotides (one of A, G, T, or C). Half of these are examples of a gene-starting sequence called a promoter (Towell et al., 1990). There are no missing values.

The *splice-junction* data set is another DNA sequence database (Noordewier et al., 1991). It consists of 3190 sequences of 60 nucleotides each. Only 0.03% of the feature values are missing. There are three categories. Two of them represent different boundary regions between protein-coding sections (exons) and non-coding sections (introns) of a DNA sequence. The third category is `negative`.

4 Experiments

4.1 Method

In order to compare the performance of the three systems, learning curves were generated for each of the five data sets. Each system was trained in batch fashion on increasingly

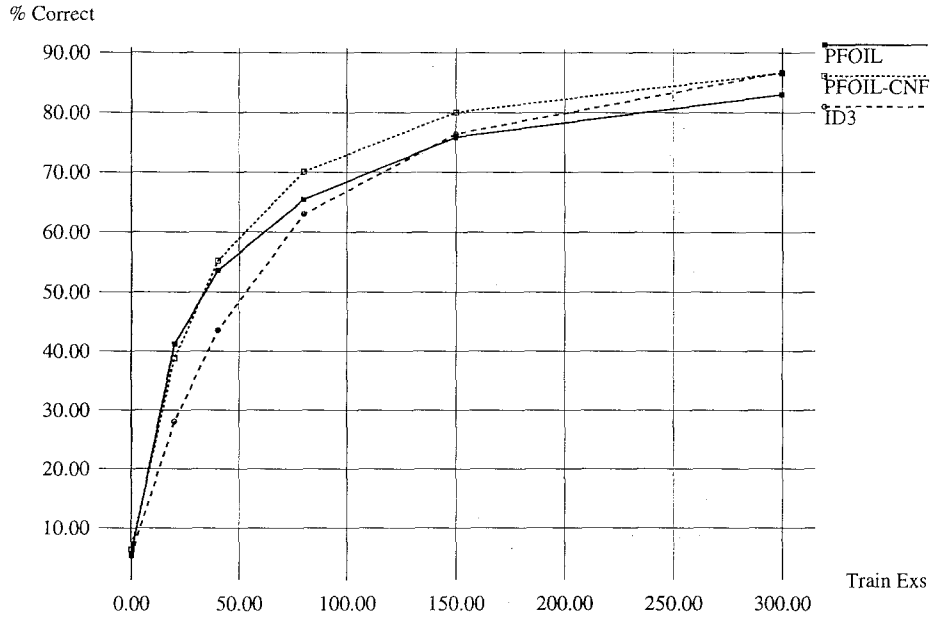

Figure 1. Test accuracy for soybean data.

larger fractions of a fixed training set and repeatedly tested on the same disjoint test set. At each point, the following statistics were gathered: training set accuracy, test set accuracy, training time, and concept complexity. For concept complexity, we recorded the number of literals for CNF and DNF formulae and the number of leaves for decision trees. Although, literal and leaf counts are not directly comparable, they provide a reasonable measure of relative complexity.

All of the results were averaged over 25 trials, each with a different randomly selected training and test set. In most domains, the test set consisted of all remaining examples; however, a random set of 200 test examples was used for the larger splice-junction data. The results were statistically evaluated using a two-tailed paired t-test. For each training set size, each pair of systems was compared to determine if their differences in accuracy, time, and complexity were statistically significant ($p < 0.05$).

4.2 Results

Representative curves for test accuracy, concept complexity, and train time for some of the domains are shown in Figs. 1–7. Run times are for a SPARCstation 2 running Lucid Common Lisp. Due to the specifics of the implementations, when given zero training examples, PFOIL classifies everything as negative, PFOIL-CNF classifies everything as positive, and ID3 returns a random class. Differences at zero training examples are not particularly meaningful and will not be discussed. If other specific differences are not mentioned, they should be assumed to be statistically insignificant. When orderings of systems are given, it is always from "best" to "worst."

On the soybean data, PFOIL-CNF performed quite well. Its accuracy was significantly greater than PFOIL's after 80 examples, and significantly greater than ID3's except at 300

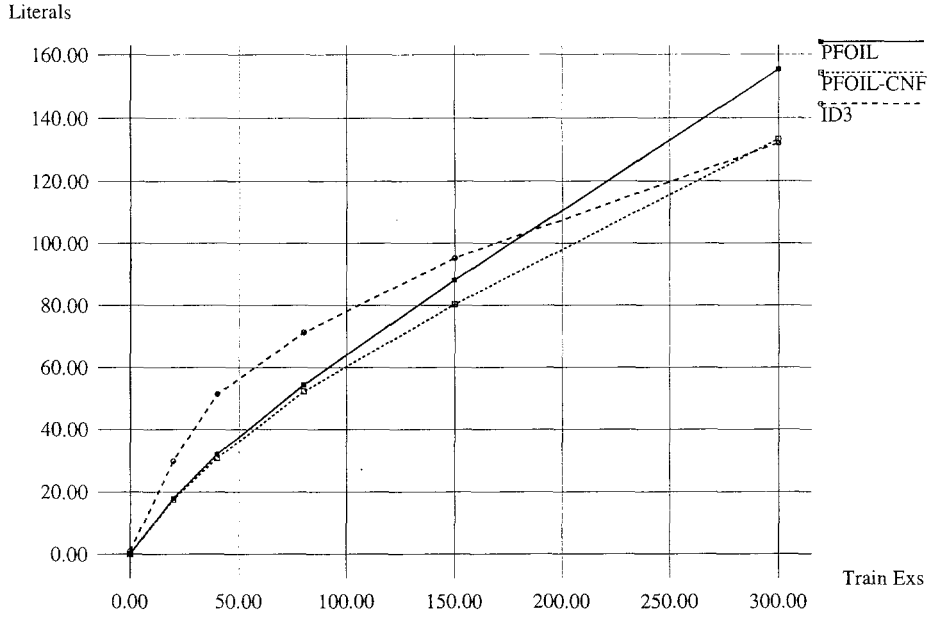

Figure 2. Concept complexity for soybean data.

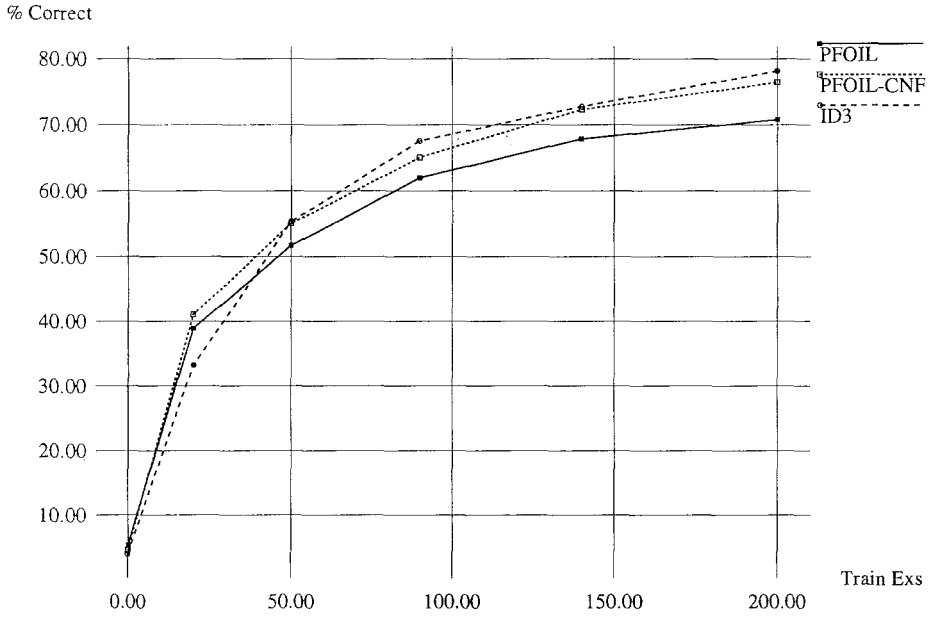

Figure 3. Test accuracy for audiology data.

examples. PFOIL was significantly better than ID3 at 20 and 40 examples and significantly worse at 300 examples. With respect to training time, all differences after 80 examples were significant, with the ordering: ID3, PFOIL-CNF, PFOIL. Regarding concept complexity,

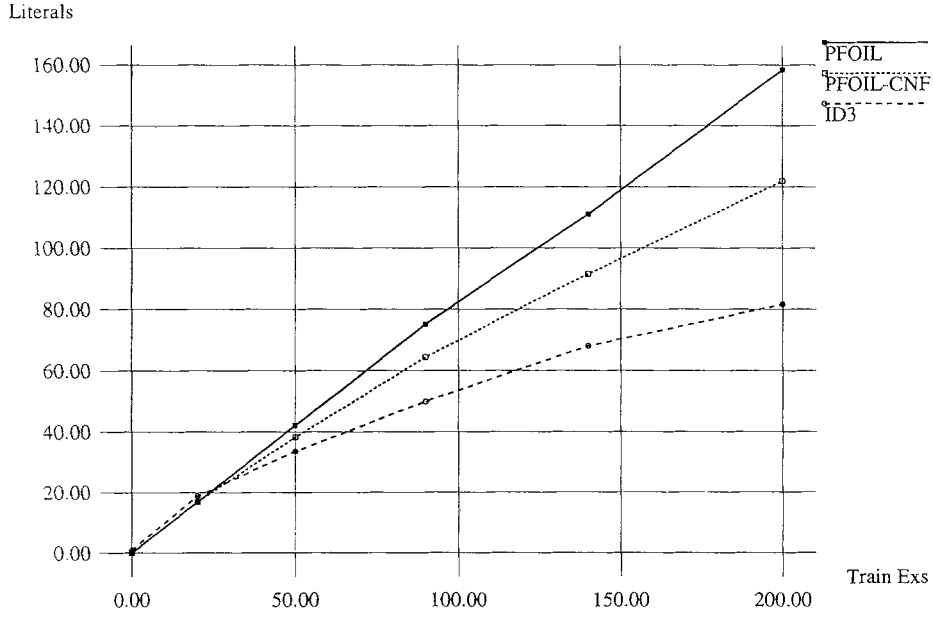

Figure 4. Concept complexity for audiology data.

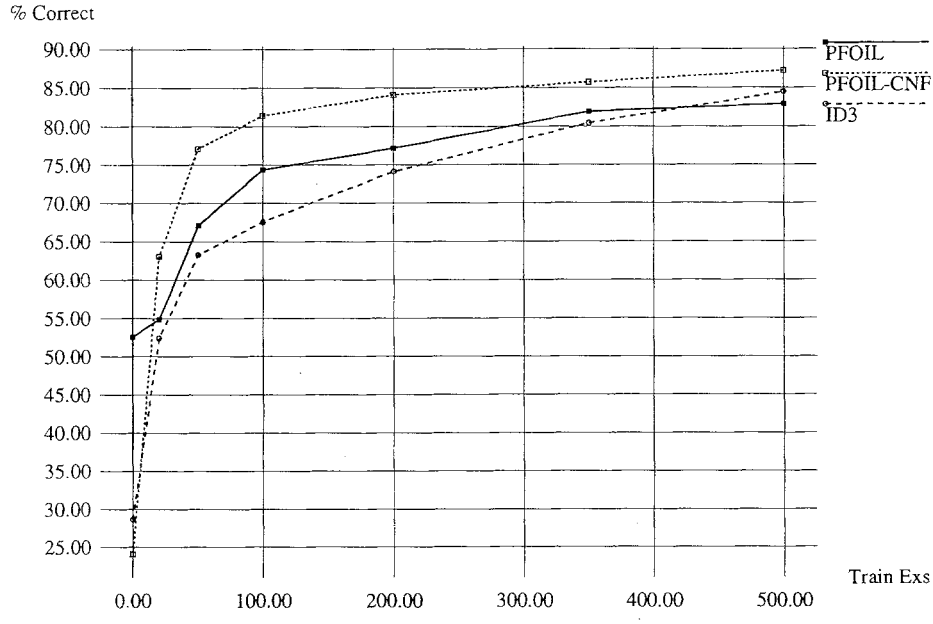

Figure 5. Test accuracy for splice junction data.

PFOIL-CNF always produced simpler results than PFOIL. Comparing leaf and literal counts is problematic, but ID3 did significantly worse than the other two except at 300 examples.

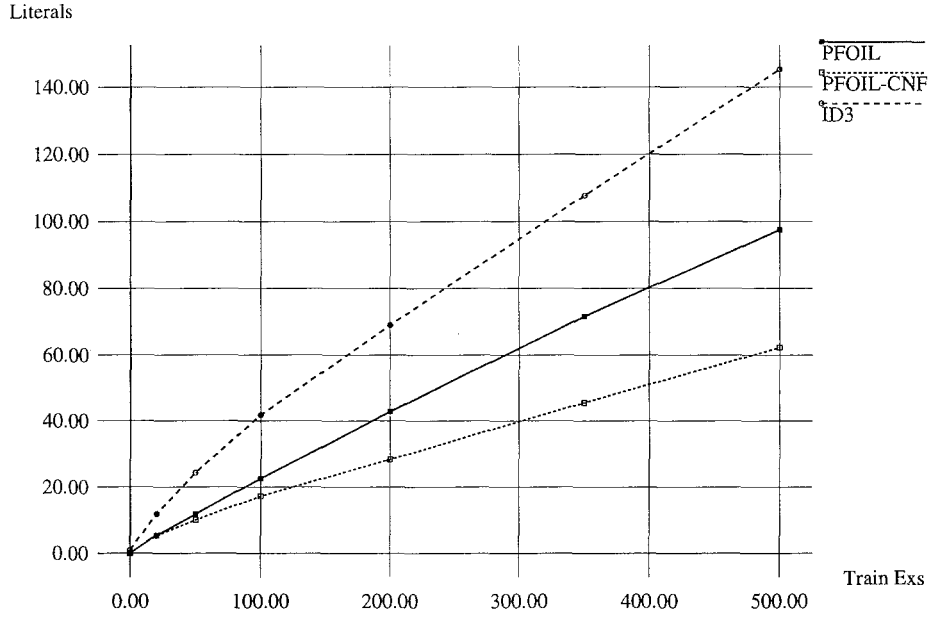

Figure 6. Concept complexity for splice junction data.

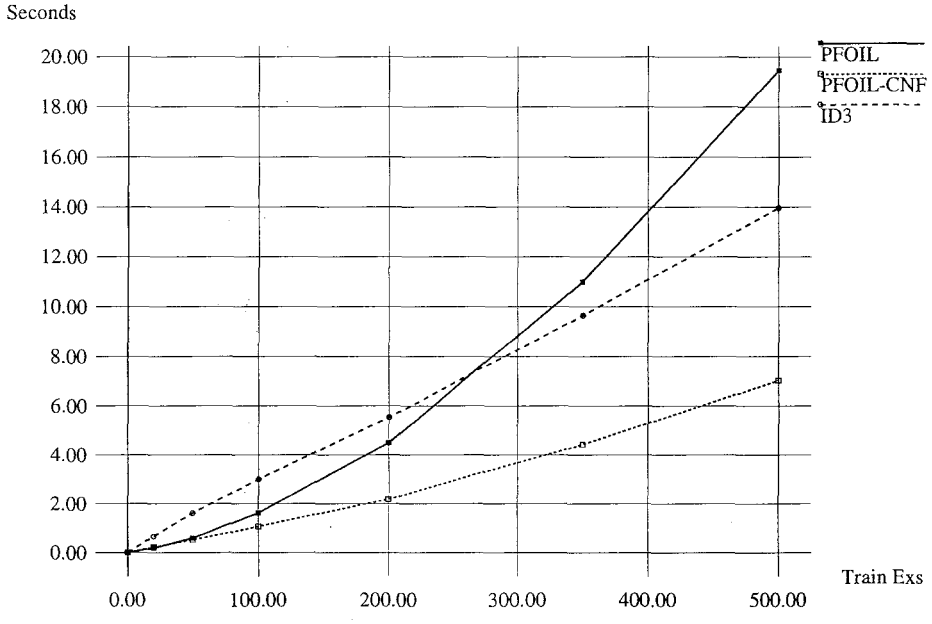

Figure 7. Training time for splice junction data.

On the congressional voting data there were no striking differences. The graphs are not shown; however, data was gathered for 20, 50, 100, 180, and 300 training examples. The only significant difference in accuracy was that PFOIL did slightly worse than the other two

at 300 examples. All training times after 50 examples were significant, with the ordering: PFOIL, PFOIL-CNF, ID3; however, the difference between PFOIL and PFOIL-CNF was minimal. All differences in concept complexity were significant with the ordering: ID3, PFOIL, PFOIL-CNF; however, the difference between PFOIL and PFOIL-CNF were again minimal.

On the audiology data, CNF showed some advantages. Regarding accuracy, PFOIL-CNF was better than ID3 at 20 examples and better than PFOIL at 140 and 200 examples. PFOIL was better that ID3 at 20 examples but consistently worse after 90 examples. All training times were significantly different, generally with the ordering: PFOIL-CNF, PFOIL, ID3; except that ID3 was faster than PFOIL at 200 examples and PFOIL was faster than PFOIL-CNF at 20 and 50 examples. All differences in concept complexity after 50 examples were significant with the ordering: ID3, PFOIL-CNF, PFOIL; but ID3 was significantly worse than the other two at 20 examples.

On the promoter data, CNF again showed a slight advantage. The graphs are not shown; however, data was collected for 10, 20, 40, 60 and 80 training examples. The results are qualitatively similar to the splice junction results but the differences are smaller. With respect to accuracy, PFOIL-CNF was significantly better than PFOIL at 40 examples and better than ID3 at 40 and 60 examples. All training times were significantly different with the ordering: PFOIL-CNF, PFOIL, ID3; except that there was no significant difference between PFOIL and PFOIL-CNF at 40 and 60 examples. All differences in concept complexity were also significant with the ordering: PFOIL-CNF, PFOIL, ID3; except there was no significant difference between PFOIL and PFOIL-CNF at 10 and 20 examples.

On the splice-junction data, CNF shows a fairly strong advantage. On accuracy, PFOIL-CNF is consistently better than the other two systems, with differences as large as ten percentage points. PFOIL is significantly better than ID3 at 100, 200, and 300 examples but significantly worse than ID3 at 500. On training time, PFOIL-CNF is again consistently better, except it is slower than PFOIL at 20 examples. PFOIL is significantly faster than ID3 up to 200 examples, but significantly slower for 500 examples. All differences in concept complexity are also significant with the ordering: PFOIL-CNF, PFOIL, ID3; except for PFOIL and PFOIL-CNF at 20 examples.

4.3 Discussion

Overall, the CNF-learner performed surprisingly well. With respect to classification accuracy, it was *never* statistically significantly worse than the other two systems and was frequently superior. In addition, PFOIL-CNF ran faster and produced simpler concepts than PFOIL, except on the congressional voting data, where the difference was minimal (less than 10%). Except on the soybean data, PFOIL-CNF also ran faster than ID3. It also produced simpler concepts in three of the five domains, assuming literal and leaf counts are comparable measures.

Since all of the algorithms are "top down" and "grow" concept descriptions until they are consistent with the training data, their run time is a function of their concept complexity. Consequently, PFOIL-CNF's faster run time is explained by the fact that it is learning simpler concepts. The fact that PFOIL-CNF frequently learned more accurate, simpler concepts indicates that CNF was a somewhat better bias for the domains tested. Apparently, the assumption that the concepts in these domains could be represented as simple CNF formulae was frequently slightly more appropriate than the assumption that they could be represented as simple DNF formulae or simple decision trees.

Table 3. Sample DNF and CNF concepts in the splice-junction domain.

DNF for Intron-Exon border (26 literals)

```
P-2=A ∧ P-1=G ∧ P-9=T ∧ P-7=C ∨
P-2=A ∧ P-1=G ∧ P-21=C ∧ P-25=C ∨
P-2=A ∧ P-1=G ∧ P2=G ∧ P-3=C ∨
P-2=A ∧ P-1=G ∧ P-10=T ∧ P-14=T ∨
P-2=A ∧ P-1=G ∧ P-8=C ∧ P-20=T ∨
P19=A ∧ P10=C ∧ P-13=C ∨
P-21=G ∧ P-19=A ∧ P-26=A
```

CNF for Intron-Exon border (17 literals)

```
P-2=A ∨ P3=T ∧
P-1=G ∨ P20=C ∧
P-3=C ∨ P-10=T ∨ P23=A ∧
P-7=C ∨ P-21=C ∨ P-19=T ∨ P25=A ∨ P12=A ∧
P-18=C ∨ P-15=C ∨ P6=G ∨ P16=A ∨ P-10=G
```

Table 4. Sample DNF and CNF concepts in the audiology domain.

DNF for Normal Ear (34 literals)

```
AIR=NORMAL ∧ O-AR-U=ABSENT ∧ AR-U=NORMAL ∧ MOD-GT-4K=FALSE ∨
AIR=NORMAL ∧ O-AR-C=ELEVATED ∧ NOISE=FALSE ∧ O-AR-U=ELEVATED ∨
AIR=NORMAL ∧ O-AR-C=ABSENT ∧ AR-U=ELEVATED ∧ TYMP=A ∨
AIR=NORMAL ∧ NOISE=FALSE ∧ AGE-GT-60=FALSE ∧ BONE=UNMEASURED ∧ O-AR-U=ELEVATED ∨
AIR=NORMAL ∧ NOISE=FALSE ∧ O-AR-C=ELEVATED ∧ SPEECH=VERY-GOOD ∨
AIR=NORMAL ∧ SPEECH=NORMAL ∧ AR-C=ELEVATED ∧ NOTCH-AT-4K=FALSE ∨
AIR=NORMAL ∧ SPEECH=NORMAL ∧ NOISE=FALSE ∧ NAUSEA=FALSE ∧ AR-U=NORMAL
  ∧ DIZZINESS=FALSE ∧ AGE-GT-60=FALSE ∧ NOTCH-AT-4K=FALSE ∧ AR-C=NORMAL
```

CNF for Normal Ear (13 literals)

```
AIR=NORMAL ∧
NOISE=FALSE ∧
O-AR-U=ABSENT ∨ O-AR-C=ELEVATED ∨ SPEECH=NORMAL ∧
M-SN-GT-2K=FALSE ∧
M-SN-GT-4K=FALSE ∧
TYMP=A ∧
DIZZINESS=FALSE ∨ O-AR-C=ELEVATED ∧
NOTCH-AT-4K=FALSE ∧
M-SN-2-3K=FALSE ∧
MOD-GT-4K=FALSE
```

Specific examples of DNF and CNF concepts learned from the same data are given in Tables 3 and 4. Notice that the DNF formulae contain repeated patterns of literals across disjuncts. The DNF results for the soybean and promoter domains show similar but less pronounced replication. Since representing a CNF concept in DNF generally requires such duplication, this is another indication that CNF may be a superior bias in these domains. Pagallo and Haussler (1990) discuss similar duplication resulting from representing DNF formulae as decision trees.

It should be noted that using intervals on linear features and internal disjunction (allowing disjunctions of values for the same feature in the "literals" of a DNF formula) (Michalski,

1983), can help relieve the replication problem for DNF. However, most of the clauses in the sample CNF formulae are not amenable to either of these approaches.

Unlike DNF formulae, decision trees can "share" conceptual structure across multiple disjuncts and avoid some of the duplication problems discussed above. However, decision trees have their own replication problem in which subtrees must be duplicated in order to represent certain disjunctions (Pagallo & Haussler, 1990). An examination of a number of decision trees produced for the current data sets revealed a few, small, duplicated subtrees. However, replication was more clearly indicated by the repeated use of the same features in different parts of the trees.

Unfortunately, the differences between decision trees and DNF/CNF formulae make direct comparisons of these representations somewhat difficult. In particular, a single decision tree can discriminate any number of categories, while separate DNF and CNF formulae are needed for each category. This allows decision trees to share conceptual structure across multiple categories in a way that CNF and DNF representations cannot. This explains why ID3 learns less complex concepts in the audiology domain, since it has the largest number of categories (24) and therefore the greatest opportunity for sharing structure across categories.

5 Related research

As previously mentioned, there has been very little experimental research on learning CNF. Some extensions to the FRINGE (Pagallo & Haussler, 1990) and CITRE (Matheus & Rendell, 1989) work on constructive induction in decision trees include feature-construction heuristics appropriate for learning CNF. Pagallo (1990) explored a dual version of FRINGE which constructs disjunctive features from negative leaves and a symmetric version which constructs features appropriate for both DNF and CNF. Yang et al. (1991) present an improved version of symmetric FRINGE called DCFRINGE. However, unlike PFOIL-CNF, these systems do not directly construct CNF formulae in a single pass. Also, the authors do not present results comparing CNF and DNF biases on natural data.

6 Future research issues

Decision-tree methods for discretizing continuous attributes (Quinlan, 1986; Fayyad & Irani, 1992) could be employed to handle real-valued features. The effect of using numerical thresholds and internal disjunction in DNF formulae needs to be determined. Also, the relative effects of pruning on different representations should be examined.

As previously mentioned, for some purposes it is difficult to compare multi-category, n-ary decision trees to DNF and CNF formulae. A learner that induces a separate decision tree for each category and is restricted to binary splits on specific feature values would be easier to compare. Factoring out these additional differences might allow the relative advantages and disadvantages of decision tree representations to be studied more clearly.

Although it did not arise frequently in the current experiments, CNF, not surprisingly, has its own replication problem. An example from the current experiments is shown in Table 5, in which the literal `STEM-CANKERS=ABOVE-SEC-NDE` is repeated in six clauses.

Table 5. Example of replication in CNF.

CNF for Frog Eye Leaf Spot (28 literals)

```
FRUIT-SPOTS=COLORED ∨ LEAFSPOT-SIZE=>-1/8 ∧
EXTERNAL-DECAY=FIRM-AND-DRY ∨ LEAF-SHREAD=ABSENT ∧
EXTERNAL-DECAY=FIRM-AND-DRY ∨ TEMP=NORM ∨ STEM-CANKERS=ABOVE-SEC-NDE ∧
FRUIT-PODS=DISEASED ∨ SEED-TMT=FUNGICIDE ∨ HAIL=NO ∨ AREA-DAMAGED=SCATTERED ∧
STEM-CANKERS=ABOVE-SEC-NDE ∨ PLANT-GROWTH=NORM ∧
STEM-CANKERS=ABOVE-SEC-NDE ∨ SEED=NORM ∧
STEM-CANKERS=ABOVE-SEC-NDE ∨ DATE=8 ∨ DATE=9 ∨ DATE=10 ∨ HAIL=NO ∧
STEM-CANKERS=ABOVE-SEC-NDE ∨ GERMINATION=80-89% ∨ DATE=9 ∨ AREA-DAMAGED=LOW-AREAS ∨
   PRECIP=NORM ∧
STEM-CANKERS=ABOVE-SEC-NDE ∨ PLANT-STAND=NORMAL ∨ CROP-HIST=SAME-LST-SEV-YRS
```

However, even in this case, the learned DNF representation was more complex (33 literals versus 28). There are obviously concepts that can be represented more compactly in DNF than CNF; however, these did not seem to arise in the current experiments.

A standard representation that can completely avoid replication is multi-level Horn-clause theories. By appropriately introducing new intermediate terms, replication can be prevented. Inverse resolution operators like intra-construction and inter-construction (Muggleton, 1987) introduce new terms in order to remove redundancy and to compact concept representations. However, there has been little research on efficient, oracle-free methods for inducing multi-level Horn-clause theories. The symmetric FRINGE and DCFRINGE algorithms discussed above are possible exceptions but typically require many passes through the data. An integration of PFOIL and PFOIL-CNF seems like an alternative promising approach.

7 Conclusions

This paper has presented experiments comparing inductive concept learning using three different representations: DNF formulae, CNF formulae, and decision trees. In four out of five domains, the CNF learner produced more accurate concepts than the other two learners. In most cases, it also ran faster and produced simpler concepts. This is an indication that CNF may be a useful bias for real-world problems. However, there has been little focus on developing practical algorithms for learning CNF. The promising results reported in this paper will hopefully encourage more research in this and related areas.

Acknowledgments

I would like to thank the editor, Dennis Kibler, and the anonymous reviewers for their helpful comments on the initial draft of this paper. I would also like to thank John Zelle for stimulating discussions that inspired the development of PFOIL-CNF. This research was supported by the National Science Foundation under grant IRI-9102926 and the Texas Advanced Research Program under grant 003658114.

Notes

1. One can easily show that the time complexity of both PFoil and PFoil-CNF is $O(nec)$ where n is the number of possible literals, e is the number of examples, and c is the complexity of the final learned concept in number of literals (i.e. adding each literal in the definition requires determining the gain of each possible literal which requires examining each remaining example). The complexity of the learned concept, c, is in turn bounded by $O(ne)$, since each term in a DNF formula has some subset of the n possible literals and covers at least one new positive example in order to have positive gain (a dual argument holds for CNF). Therefore, both algorithms are $O(n^2e^2)$.

References

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Buntine, W., & Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8(1):75–86.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–284.

Fayyad, U.M., & Irani, K.B. (1992). On the handling of continuous-valued attributes in decision-tree generation. *Machine Learning*, 8(1):87–102.

Matheus, C.J., & Rendell, L.A. (1989). Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, 645–650.

Michalksi, R., Mozetic, I., Hong, J., & Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1041–1045.

Michalski, R.S. (1975). Synthesis of optimal and quasi-optimal variable valued logic formulas. In *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, Bloomington, IN, 76–87.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell, (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 83–134.

Michalski, R.S., & Chilausky, S. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Journal of Policy Analysis and Information Systems*, 4:126–161.

Muggleton, S. (1987). Duce, an oracle based approach to constructive induction. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 287–292.

Murphy, P.M., & Aha, D.W. (1993). UCI repository of machine learning databases (machine-readable data repository at ics.uci.edu). Department of Information and Computer Science, University of California, Irvine, CA.

Noordewier, M.O., Towell, G.G., & Shavlik, J.W. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems*, 3, San Mateo, CA: Morgan Kaufman.

Pagallo, G. (1990). *Adaptive Decision Tree Algorithms for Learning from Examples*. PhD thesis, University of California, Santa Cruz, CA.

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

Pitt, L., & Valiant, L.G. (1988). Computational limitations on learning from examples. *Journal of the Association for Computing Machinery*, 35(4):965–984.

Porter, B., Bareiss, R., & Holte, R. (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263.

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

Quinlan, J.R. (1989). Unknown attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY, 164–168.

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

Schlimmer, J.C. (1987). *Concept Acquisition Through Representational Adjustment*. PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA.

Shavlik, J.W., Mooney, R.J., & Towell, G.G. (1991). Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143.

Towell, G.G., Shavlik, J.W., & Noordewier, M.O. (1990). Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, 861–866.

Valiant, L.G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142.

Yang, D., Rendell, L., & Blix, G. (1991). A scheme for feature construction and a comparison of empirical methods. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 699–704.

Received November 5, 1992
Accepted May 4, 1993
Final Manuscript November 19, 1993