Reducing the small disjuncts problem by learning probabilistic concept descriptions

Kamal M Ali
ali@ics.uci.edu
Michael J Pazzani
pazzani@ics.uci.edu

Technical Report 92-111

December 17, 1992

Reducing the small disjuncts problem by learning probabilistic concept descriptions

Kamal M Ali Michael J Pazzani

This paper presents a method for learning relational and attribute-value concepts based on maximum-likelihood estimation. Greedy hill-climbing classifiers like FOIL and FOCL build a few reliable clauses but many unreliable clauses, referred to as small disjuncts. Small disjuncts are a major source of error on independent test examples. We introduce the system HYDRA which learns probabilistic relational concepts and reduces contribution of error from small disjuncts. We demonstrate the reduction of the small disjuncts problem on various relational and attribute-value domains.

1. Introduction

Concept learners that form DNF concept descriptions have been shown to be prone to the small disjuncts problem (Holte *et al.* , 1989). This is the problem where a large proportion of the overall error on an independent training set can be attributed to clauses[1] that were learned using a small number of positive examples. In noisy domains, DNF concept learners typically learn a few reliable clauses (large disjuncts) that cover many positive examples and many clauses (small disjuncts) that cover few positive examples.

This paper presents the system HYDRA which builds probabilistic relational DNF concept descriptions that reduce the effect of small disjuncts. Probabilistic relational

1. We will refer to clauses and disjuncts interchangeably.

target concepts are those that do not have necessary or sufficient conditions for the target concept. For example, the concept C(X,Z) whose definition is given below, does not have a necessary or sufficient concept description in predicate logic.

$$p(C(X,Z)\,|\,E(X,Y),F(Y,Z)) = 0.8$$

$$p(C(X,Z)\,|\,\sim(E(X,Y),F(Y,Z))) = 0.1\ .$$

First order concept learners whose concept language does not admit probabilistic expressions will not learn a compact description for such concepts. Instead as the number of training examples presented increases, they will form more and more complex approximations to the target concept. HYDRA builds concept descriptions by attaching likelihood ratios to each clause. HYDRA aims to estimate the reliability of a clause and then uses that to reduce the effect of small disjuncts. In Section 1.1 we briefly explain how the relational concept learner FOIL learns from data. Section 1.2 compares HYDRA to previous work on the small disjuncts problem. Section 1.3 shows that FOIL has a small disjuncts problem. HYDRA is derived from FOIL (Quinlan, 1990), so Section 2 presents HYDRA and the semantics associated with our formulation of clause reliability. We show HYDRA learns concept descriptions that have low error rates when compared to FOIL and other systems. Furthermore, we present evidence in Section 3 that this increase in accuracy is due to reducing the contribution of error from small disjuncts.

1.1 FOIL

FOIL builds a concept description that is a conjunction of Horn clauses[2]. It uses a set of presupplied background relations to build a concept description such that

2. Some authors prefer to view such concept descriptions as DNF. Strictly speaking, the body of the description $(a \wedge b \rightarrow c) \wedge (d \wedge e \rightarrow c)$ is the DNF expression $(a \wedge b) \vee (d \wedge e)$.

each positive training example[3] of the target relation is covered by one or more clauses and no negative example is covered by any clause. An example for a relation consists of an ordered sequence of ground terms. We will refer to each term as a component. FOIL uses a separate and conquer approach that involves filtering out any examples covered by the current clause before presenting the remaining examples for learning of the next clause. This iterative process terminates when all positive training examples have been covered by at least one clause.

Each clause is started in the following manner: for each background relation FOIL builds candidate literals consisting of the name of that background relation and a subset of the variables in the head of the clause. So if the target relation is $P(V_1,\ldots,V_n)$ and FOIL is considering the background relation B which has arity m, FOIL will consider all literals $B(X_1,\ldots,X_m)$ where $X_1,\cdots X_m \subseteq V_1,\cdots V_n$.[4] We will refer to the sequence $(X_1,\cdots,X_m)$ as a variablization.

FOIL ranks each candidate literal by measuring the information that would be gained if that literal was conjoined with the rest of the clause. If p denotes the number of positive examples covered by the clause before conjoining with the candidate literal, p^+ the number of positive examples after the literal, n the number of negatives before the literal and n^+ the number of negatives after the literal, the information gain attributed to that literal is:

$$\text{Information-gain}(p,n,p^+,n^+) = p^+[\text{information}(p^+,n^+) - \text{information}(p,n)]$$

3. We will use the terms example and tuple interchangeably.
4. Literals where more than one of the X_i map to the some V_j are included in this framework.

where

$$\text{information}(a,b) = \lg\left(\frac{a}{a+b}\right)$$

After considering each variablization of each background relation, the best candidate literal is conjoined to the clause. The positive and negative examples that satisfy the new clause are then presented for learning the next literal. If no negative examples are covered by the new clause, FOIL removes the positive examples covered by that clause before proceeding to learn the next clause.

An important point to note about the information gain heuristic is that it trades off generality (the p^+ term) versus the log of a function that aims to maximize discriminability. Without the p^+ term, the learner could build one clause per positive training example, satisfactorily covering all the positive examples and none of the negative examples, yet obtaining a concept description with very little generalizability.

HYDRA uses this same separate and conquer control strategy but learns a concept description for each clause and attaches weights in the form of likelihood ratios to each clause[5]. We discuss these ratios and HYDRA in detail in Section 2.

1.2 Previous Work

The work of Holte *et al.* (1989) demonstrates that various attribute-value learning systems on several domain are prone to the small disjuncts problems. Our work indicates that FOIL is also prone to the small disjuncts problem on various relational and attribute-value domains. One possible explanation for the fact that the error

5. HYDRA is derived from FOIL via FOCL. FOCL (Pazzani and Kibler, 1991) added to FOIL the ability to use a prior domain theory.

contribtion of small disjuncts is greater than the error contribution of large disjuncts may simply be attributable to the relatively large number of small disjuncts learned. Our analysis compensates for this factor by comparing the proportion of errors of comission made by a set of clauses compared to the proportion of test examples matched by that set of clauses. We find that clauses that match only a small proportion of the test examples are responsible for a much larger proportion of the total comissive error (Figure 1).

Our experiments over several noisy and non-noisy domains using FOIL indicate that the propotion of errors made by clauses covering p or fewer examples is much larger than the proportion of instance space covered by those clauses. Our hypothesis then is that the small disjuncts problems can be addressed by assigning lower weights to small disjuncts.

One obvious approach that Holte *et al.* rejected was to eliminate disjuncts covering fewer than a prescribed number of examples because such an approach may delete significant small disjuncts. A disjunct is significant if it is part of the target concept description. They also rejected approximations to statistical significance tests in favor of a variable bias system that builds maximally general clauses to cover large disjuncts and maximally specific clauses to cover small disjuncts. They categorize the information gain bias (used in ID3 and CN2, for example) as a "maximum generality" bias and show that using a maximum specificity bias to learn disjuncts covering fewer than a preset threshold lowers the error rates of small disjuncts. They test their method on the KPa7KR chess endgame task and find that by using a switchover threshold of 9 examples, the overall error rate is reduced from 7.2% to

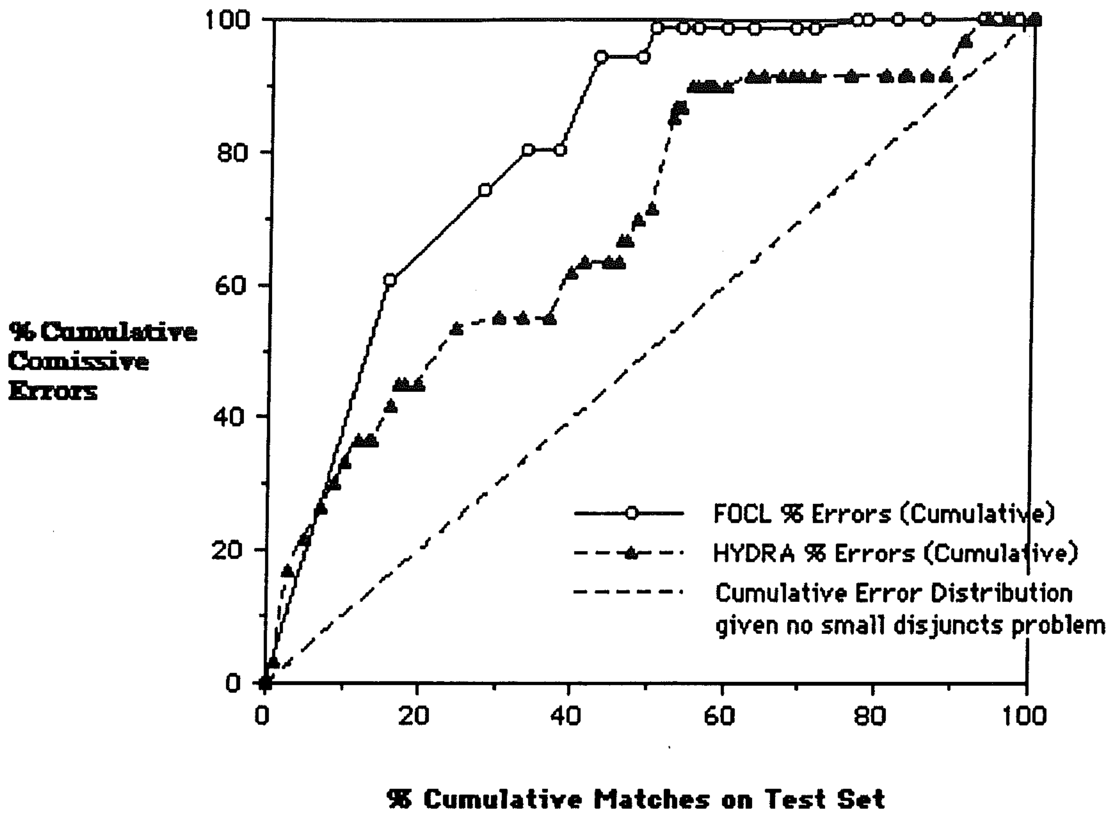

Figure 1. Clauses from FOIL that match only 20% of examples from the test set cause much more than 20% of the errors of comission. The problem is reduced for HYDRA. Both algorithms were trained on 160 examples from the illegal chess domain with 5% tuple noise added. A level of 5% tuple noise means the probability of assigning a random value[6] to a component of a tuple is 0.05. Each curve represent the average of 20 independent trials.

5.2% and that using a switchover threshold of 5 examples, the overall error rate stays at 7.2% although the error rate of small disjuncts is reduced from 16% to 11%. We have chosen to address the small disjuncts problem using a maximum-likelihood estimation approach using weights which is described in Section 2.

6. The value is randomly selected from a uniform distribution over the set of legal values for that component.

1.3 Evidence Of The Small Disjuncts Problem

Following the definition of the small disjuncts problem as given in Section 1.2, Figure 1 illustrates the correlation between cumulative matches over the test set and the cumulative comissive error distribution for FOIL on the illegal chess domain (Muggleton *et al,* 1989). This pattern is repeated without failure on several non-noisy and noisy domains and on both artificial and natural data sets such as the DNA promoters data set. If FOIL had no small disjuncts problem, it would have an cumulative error curve similar to the ideal error distribution line shown in Figure 1.

2. Probabilistic Relational Concept Descriptions

Our motivation for developing HYDRA derives from two considerations. Firstly, "real world domains" embody a variety of problems such as noisy relations and noisy class labels, so learners that aim to build necessary and sufficient concept des criptions end up overfitting the data. Secondly, as Holte *et al.*, have shown, the error contributed by small clauses is disproportionate in the sense that such clauses may be responsible for 80% of the overall comissive errors but match only 20% of the test examples.

2.1 Knowledge Representation and Classification

Here we discuss the knowledge representation used to represent concepts and how they are used in classification. The method of learning such concepts is presented in Section 2.2. A concept is represented as a conjunction of Horn clauses. Each clause has an associated weight representing the *degree of logical sufficiency* [7] (Duda, Gaschnig & Hart, 1979) which that clause has for the target concept. A couple of

7. We will abbreviate this as LS.

such clauses for different classes are shown below.

$$a(X,Y) \wedge b(Y,Z) \rightarrow Class_i(X,Z) \ [LS = 3.5]$$

$$c(X,Y) \wedge d(Y,Z) \rightarrow Class_j(X,Z) \ [LS = 4.0]$$

For $clause_{ij}$ of $Class_i$, given positive training examples ζ^+ and negative training examples ζ^-, the LS (degree of logical sufficiency) is estimated as follows.

$$ls_{ij} = \frac{P(clause_{ij}=true|\zeta^+)}{P(clause_{ij=}=true|\zeta^-)}$$

Given a test example such as the following:

$$a(p,q) \wedge b(q,r) \wedge c(i,j) \wedge d(j,k) \wedge ....$$

classification proceeds as follows. For each class, we want to estimate the probability that that test example belongs to the class given that it has satisfied some clause of that class. In order to do this, considering only clauses that are satisfied by the current test example, for each class, we choose the clause with the highest LS value. The clause with the highest LS value is chosen because it is most indicative of the class. We do this for each class and assign the test example to the class whose representative clause has the highest LS value[8]. We will refer to this as *optimistic likelihood estimation.* Another strategy for evaluating the degree to which satisfaction of the clauses indicates membership in the class is to multiply together the LS values of all the clauses within each class, the product being taken over clauses that are satisfied by the test example. We will refer to this as *pessimistic likelihood estimation.* It assumes all the clauses are independent, given the data. Both these meth-

8. If the example satisfies no clause of any class, HYDRA guesses and assigns the test example to the class that occurred with greatest frequency in the training data.

ods are empirically compared in Table 1.

2.2 Learning in HYDRA

FOIL aims to learn necessary and sufficient concept descriptions. HYDRA differs from FOIL in three major ways. Firstly, HYDRA learns a concept description for each class[9]. Secondly, HYDRA associates an estimate of the degree of logical sufficiency with each learned clause. Thirdly, HYDRA uses a candidate literal ranking metric that is aimed at learning probabilistic concept descriptions, rather than using the information gain metric which is aimed at learning necessary and sufficient concept descriptions. Learning a concept description per class is necessary when concept descriptions are going to compete to classify the test example.

HYDRA uses the same separate and conquer strategy used in FOIL. It forms clauses iteratively, removing examples covered by previous clauses in order to learn subsequent clauses. HYDRA uses a different metric to rank candidate literals than that used in FOIL. We define the LS-content of a literal covering *p* positive and *n* negative examples as follows:

$$\text{ls-content}(p, n, p_{j,0}, n_{j,0}) = \text{ls}(p, n, p_{j,0}, n_{j,0})^{1-\alpha} p^{\alpha}$$

where α is a parameter to the system, $p_{j,0}$ is the number of positive examples remaining after the previous *j-1* clauses have been built, and $n_{j,0}$ is the corresponding number of negative examples.

9. We implemented a system MC-FOCL that only differs from FOCL in that it builds a concept description for each class. This allows us to pinpoint which difference is responsible for any increase in accuracy. If clauses from more than one class claim a test example, MC-FOCL assigns the test example to the class belonging to the claiming clause which covered the largest number of positive training examples.

Using this metric HYDRA compares the LS-content before addition of the literal to that after addition of the literal. If there are no literals that cause an increase in LS-content, HYDRA completes the clause, otherwise it conjoins the literal and the current clause and resets p and n to reflect the examples that satisfy the clause with the new literal conjoined. LS-content trades off discriminability against coverage as did information content. Setting α to 0 causes HYDRA to build many clauses, none of which cover many examples. Setting α to 1 causes HYDRA to build no clauses, effectively reducing HYDRA to guessing the class with the highest prior probability estimate. For the experiments in Section 3, we set α to a neutral intermediate value of 0.5.

After all the clauses have been learned. HYDRA forms an estimate of the logical sufficiency odds multiplier ls_{ij} associated with each clause using the positive training examples ζ^+ and the negative training examples ζ^-.

$$ls_{ij} = \frac{P(clause_{ij}=true|\zeta^+)}{P(clause_{ij=}=true|\zeta^-)}$$

HYDRA estimates the numerator and denominator from the training set using the Laplace ratio (Kruskal and Tanur, 1978). According to Laplace's law of succession, if a random variable X, whose domain consists of 2 values, has been observed to take on a value v n_i times out of N trials, the least biased estimate of P(X=v) is $(n_i+1)/(N+2)$. In order to estimate ls_{ij} we note that the set of p positive examles can be split into 2 classes: those that satisfy the clause and those that do not. If p^+ of the p positive examples satisfy the clause, we can make the following estimation:

$$p(\text{clause}_{ij}=\text{true}|\zeta^+) \cong \frac{p^+ + 1}{p+2}$$

An analogous approximation can be made for the negative examples to yield

$$ls_{ij} = ls(p^+, n^+, p, n) = \frac{(p^+ + 1)/(p+2)}{(n^+ + 1)/(n+2)}$$

Note that the Laplace ratio also has the convenient property that it does not assign a LS of infinity to a clause that covers 0 negative training examples. A LS of infinity means satisfaction of that clause is totally sufficient to classify the test example as a member of the class associated with that clause. When a clause covers 0 negative examples, the expression above assigns a LS value that is the proportion of positive training examples covered by that clause multiplied by the number of negative examples that the clause managed to exclude. When comparing two clauses from the same class, each of which cover 0 negative examples the LS collapses to measuring the positive coverage of the clause.

3. Experimental Results

In this section, we show that the three changes we have made to transform FOIL into HYDRA significantly reduce prediction error rates in noisy domains although they slightly increase error rates when learning a necessary and sufficient target concept. We present evidence that HYDRA reduces the small disjuncts problem suggesting that a method that weighs unreliable clauses less heavily leads to lower error rates. We also explore the effect of varying the α parameter and present a method for reducing errors of omission made by HYDRA.

In our experiments we first compared FOIL to MC-FOIL; a system that we created to isolate the effect of learning multiple concept descriptions. Thus, MC-FOIL only differs from FOIL in that it learns one concept description for each class in the training data. If a test example matches clauses from more than one class, the test example is classified to the class whose clause covers the greater number of positive training examples. The hope is that clauses covering more positive examples are more reliable. MC-FOIL's accuracy on noisy data sets is significantly more accurate than that of FOIL. This experiment tested to see the effect that learning more than one concept description may have.

Next, we experimented to see what effect adding weights to clauses would have by comparing HYDRA (using the information gain metric) to MC-FOIL (also using the information gain metric). This change helped significantly on the promoters domain but caused an increase in error rates when learning necessary and sufficient concepts. Finally, we compared HYDRA using information gain to HYDRA using ls-content. This helped lower error significantly on domains with tuple noise and the promoters domain. It did not hurt accuracy on any domain. Altogether, these three changes work in tandem to increase classification accuracy.

3.1 Description of the domains

We ran experiments on six variants (see Table 1) of the task of predicting whether a chess board configuration was *illegal* where a board is represented as a 6-tuple consisting of the file and rank coordinates of a white king, white rook and a black king. A board is labelled *illegal* if either king is in check or the 6-tuple represents more than one piece occupying the same position. In order to form a description for *illegd*($V_1 \cdots V_6$), HYDRA uses the relations *near-file, between-file* and *equal-file,* and their rank counterparts. We also ran experiments on the "natural" domains of breast-cancer recurrence, DNA promoter and lymphography. These domains have been extensively used by attribute-value learners. Background relations for these domains consist of *equal,* <, > as well as domain-specific relations such as the *nucleotide-family* relation in the promoters domain. The last domain we studied is the King-Rook King-Pawn (KPa7KR) domain which was also used by Holte *et al.* in their study of small disjuncts.

3.2 Experimental comparison of HYDRA and FOIL

Table 1 shows that a method of assigning lower weights to less reliable clauses in noisy domains and even in the non-noisy DNA domain (where the data is not noisy but the target concept may not be expressible as a Horn theory) can yield concept descriptions with lower error rates. The *illegal* tasks with 20% class noise mean that on average, 20% of the training examples had their class labels randomly reassigned.

HYDRA's accuracy is highly competitive with other noise tolerant algorithms in all but the cancer domain. HYDRA does better than the variable bias system of Holte *et al.* on the KPa7KR domain and as well as the best algorithms[10] on lymphography. It also does better than Reduced-Error Pruning applied to FOIL (Brunk and Pazzani, 1990) and better than Reduced-Error Pruning on other domains we tested (Ali and Pazzani, 1992). Furthermore, while Holte *et al.* were not able to reduce the

10. Bayesian classifiers with options and weights (Buntine, 1990).

Task	Number of training examples	FOIL accuracy	Optimistic likelihood Estimation	Pessimistic likelihood Estimation
Illegal with 20%	160[11]	83.9 (6.6)	**91.8** (2.5)	**91.7** (3.2)
Class noise	320	83.8 (4.6)	**92.7** (4.6)	**92.5** (4.5)
Illegal with 5%	160	90.6 (5.0)	**93.6** (3.4)	**92.3** (4.0)
tuple noise	320	90.7 (3.8)	**96.5** (2.7)	**96.3** (2.3)
Noiseless Illegal	100	**97.1** (3.3)	95.1 (3.5)	93.6 (4.3)
	200	**99.1** (0.9)	96.7 (2.7)	95.9 (2.6)
Lymphography	99	**78.2**[12] (4.2)	**79.8** (5.4)	**78.6** (5.8)
KPa7KR	200	90.3 (2.5)	**94.7** (1.1)	
Breastcancer	191	63.5 (4.3)	68.9 (4.0)	**72.5** (2.2)
DNA	105	73.6 (44.3)	**81.1** (39.3)	**81.1** (39.3)

Table 1: Predictive accuracy rates of FOIL versus HYDRA. The figures in parentheses are sample standard deviations. For each task, the algorithm or set of algorithms that performed the best are in bold font. These accuracies include the "default rule" which is to guess the most frequently occuring class. For each algorithm and each task, we ran 20 independent trials, each time using the number of examples shown for training and another 50% of that number for testing. Standard deviations are high for the DNA task because we used leave-one-out testing on that domain.

overall error rate by replacing information gain with a selective specificity bias system, HYDRA is able to attain significantly lower error rates, due in part to addressing the small disjuncts problem.

Figures 1 and 2 illustrate that the small disjuncts problem is reduced by HYDRA. If an algorithm's curve goes through the point (20,80) that means that clauses matching 20% or less of the test examples made 80% of the errors of comission. Only one "match" is attributed per test example. For FOIL, this attribution is made to the

11. The training set sizes were chosen to allow comparisons with other algorithms that have been run on this domain. Examples were drawn without replacement for all but the *illegal* domain for which examples can be generated from a set of 8^6 examples. For the DNA domain, we used the leave-one-out methodology to train on 105 of the 106 available examples.
12. This accuracy is for MC-FOIL because FOCL cannot be run on domains with more than two classes and the lymphography data set contains four classes.

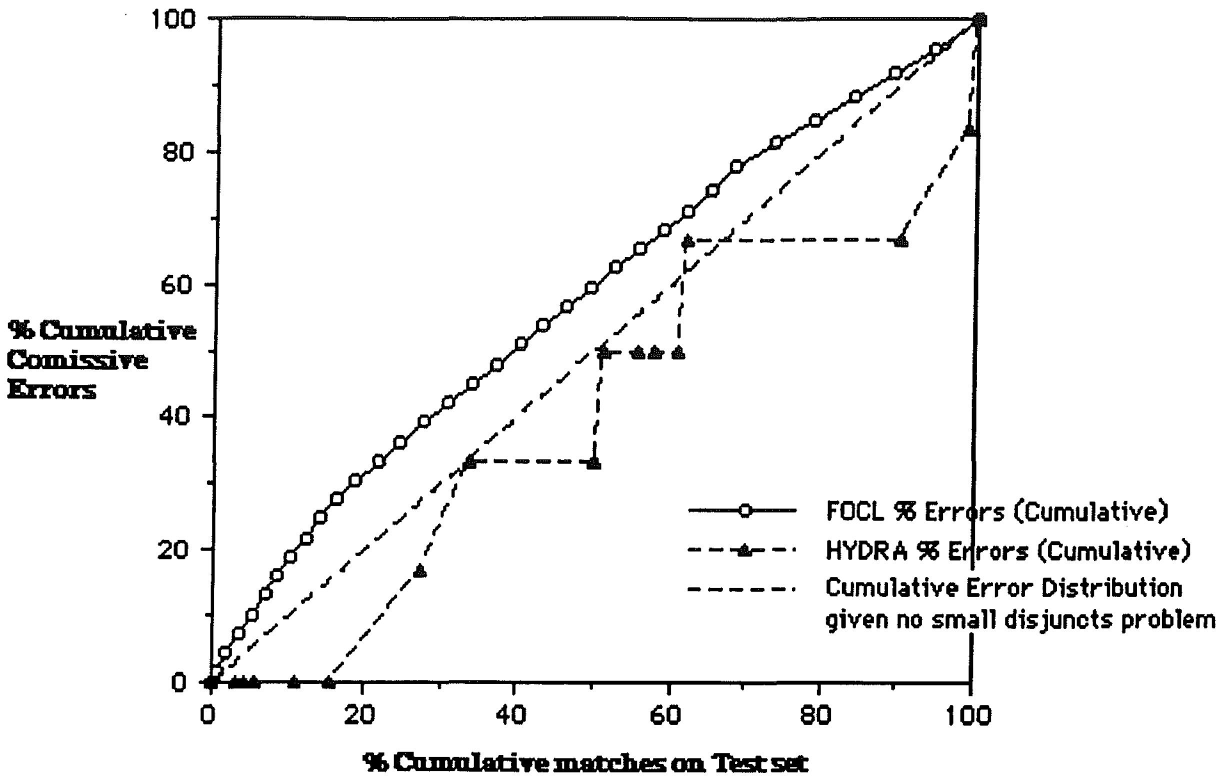

Figure 2. Comparison of the cumulative error distribution of FOIL and HYDRA to the ideal cumulative error distribution on the DNA promoters domain.

first[13] clause that is true on the test example. For HYDRA the example is attributed to the clause with the highest LS value, considering only clauses that were satisfied by the current test example. One can see from these graphs that for FOIL the clauses that match on a small percentage of the test set produce a disproportionate percentage of the total errors of comission. Note that these graphs only show the distribution of error; in particular, two algorithms may have the same distribution but very different overall error rates. In order to make a comparison between algorithms it is necessary to compare overall accuracy (Table 1) as well as the distribution of error.

13. For FOCL, clauses are examined in the order they were learned.

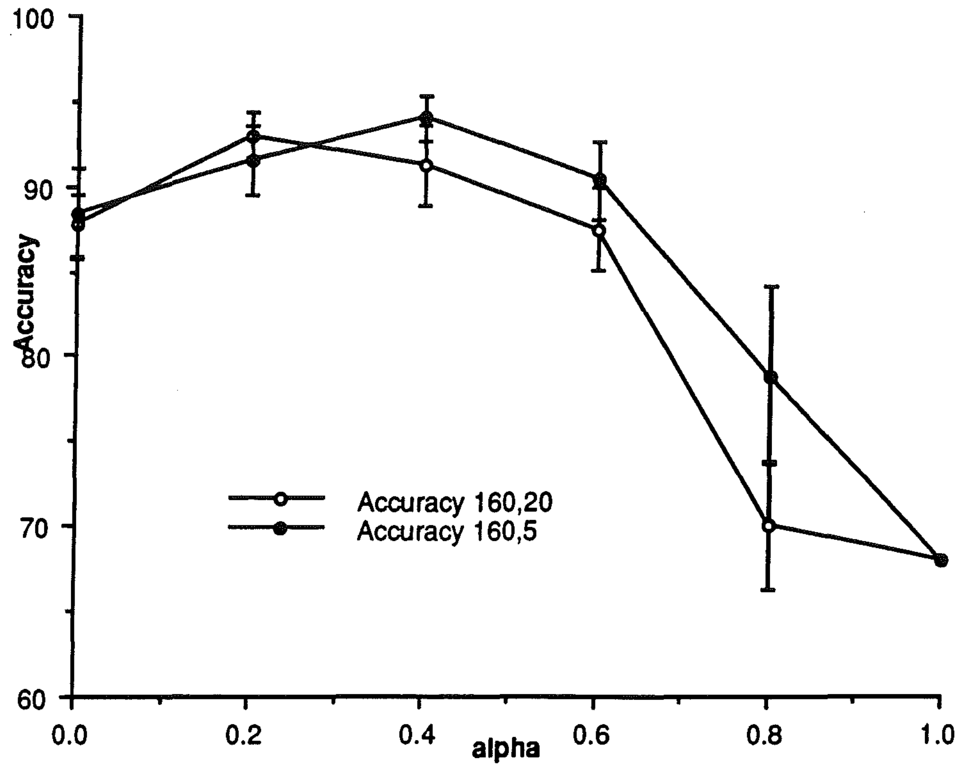

Figure 3. Variation in accuracy as a function of varying the degree to whch training data is fitted. The curve labeled "Accuracy 160,20" refers to training on 160 examples (from the King-Rook-King domain) containing 20% class noise. Accuracy 160,5 refers to training on 160 examples containing 5% tuple noise. Both curves represent averages over 15 trials. Bars correspond to one standard deviation in accuracy.

3.3. Effect of varying α

One of the major challenges in learning from noisy data is to avoid overfitting the training data. Both the information gain metric used in FOIL and the ls-content metric used in HYDRA trade off coverage against discriminability. However, this trade off is made explicit in HYDRA through the use of the alpha parameter. Figure 3 plots how varying alpha affects accuracy. Figure 3 presents preliminary evidence that the best value of alpha is one that neither over-fits the data (alpha = 0) nor one that underfits the data (alpha = 1).

Task	Accuracy without Partial clauses	Omissions Vector before Partial clauses	Accuracy with Partial clauses	Omissions Vector after Partial clauses
KRK 10	68.9	(9.1,10.7)	67.5	(1.2,0.8)
KRK 20	71.9	(5.7,5.5)	68.3	(0.1,0.6)
KRK 30	78.7	(4.5,6.0)	77.2	(0.4,1.2)
KRK 50	81.0	(3.6,4.8)	83.1	(0.0,0.1)
KRK 80	85.8	(4.9,3.8)	86.0	(0.0,0.0)
KRK 160	90.6	(3.1,1.1)	92.5	(0.0,0.0)
KRK 320	93.8	(0.5,0.3)	94.4	(0.0,0.0)
Cancer	66.6	(1.9,1.0)	67.5	(0.0,0.0)
Lymph.	81.4	(10.7,16.3,11.3,0.0)	83.5	(0,0.7,0,0)
KPa7KR	94.7	(2.5,3.3)	94.9	(0.0,0.0)
DNA	81.1	(13,21)	86.8	(0,1.9)

Table 2. Adding partial clauses helps except when the data is sparse. KRK 10 refers to data sets containing 10 training examples from the King-Rook-King domain. All KRK data sets had 20% class noise. These figures represent the averages of 20 trials.

3.4. Partial Clauses

Concept descriptions learned by HYDRA and FOIL suffer from large numbers of errors of omission when learning M of N concepts (Spackman, 1988) or when learning highly disjunctive concepts. On the DNA promoters domain for example, 13% of test examples from the promoters class, and 21% of test examples of the non-promoters class failed to match any clause of any concept description learned by HYDRA. In such cases HYDRA is forced to guess the most frequent class. A better alternative is to determine if the example nearly matches some clause. We implemented this idea by adding clauses that are derived from the clauses already learned by HYDRA. For example, the clause *a(X,Z) and b(Y,X) and c(X,X)* $\rightarrow$ *concept(X,Y)* would give rise to the following additional clauses:

a(X,Z) and b(Y,Z) $\rightarrow$ *concept(X,Y)*

a(X,Z) $\rightarrow$ *concept(Y,Z)*

These clauses serve as backup in case a test example does not satisfy any of the original clauses. These clause, which we will refer to as "partial clauses" tend to cover larger amounts of the instance space but tend to do a poorer job of discriminating positive from negative examples (and hence, have lower LS values). Table 2 gives a comparison of accuracies with and without "partial clauses"[14]. Table 2 indicates that adding partial clauses helps a lot on the DNA domain and helps to smaller extents on other natural domains. Adding partial clauses only hurts accuracy when learning from sparse data (few training examples). The omissions vector indicates the percentage of test examples that failed to match any clause of any concept description. As expected, addition of partial clauses reduces components of these vectors.

4. Conclusions and Future Work

We have presented a method using maximum likelihood estimation for reducing the small disjuncts problem and thereby increasing predictive accuracy. This method has been tested on domains requiring relational concept descriptions and those requiring attribute-value concept descriptions. We plan to extend HYDRA to build several independent[15] concept descriptions per class and then combining evidence from these models. This has been referred to as averaging multiple models (Buntine, 1991). We feel that learning multiple models will help HYDRA and further reduce the problems that hill-climbing systems like FOIL and HYDRA experience in noisy domains.

References

Ali K and Pazzani M. (1992). *HYDRA: A noise-tolerant Relational Concept Learning Algorithm* (Technical Report 92-85). Irvine, CA: University of California, Department of Information

14. Average accuracies shown in Table 2 are slightly different from those shown in Table 1 because the tables used different sets of runs of HYDRA.
15. Independent, given the training set.

and Computer Sciences.

Allen J., Thompson K. (1991). Probabilistic Concept Formation in Relational Domains. In *Proceedings of the Eighth International Workshop on Machine Learning.* Evanston, IL. Morgan Kaufmann.

Brunk C., Pazzani M. (1991). An Investigation of Noise-Tolerant Relational Concept Learning Algorithms. In *Proceedings of the Eighth International Workshop on Machine Learning* . Evanston IL. Morgan Kaufmann.

Buntine W. (1990). *A theory of learning classification rules.* Doctoral dissertation. School of Computing Science, University of Technology, Sydney, Australia.

Buntine W. (1991). Classifiers: A Theoretical and Empirical Study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Sydney, Australia: Morgan Kaufmann.

Duda R., Gaschnig J. and Hart P. (1979). Model design in the Prospector consultant system for mineral exploration. In D. Michie (ed.), *Expert systems in the micro-electronic age.* Edinburgh, England. Edinburgh University Press.

Dzeroski S. and Bratko I. (1992). Handling noise in Inductive Logic Programming. In *Proceedings of the International Workshop on Inductive Logic Programming (ILP 92).* Tokyo, Japan. ICOT.

Holte R., Acker L. and Porter B. (1989). Concept Learning and the Problem of Small Disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Detroit, MI. Morgan Kaufmann.

Kruskal W. H. and Tanur J.M. (1978). *International Encyclopedia of statistics.* New York, NY: Free Press.

Muggleton S., and Feng C. (1990). Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory.* Tokyo, Japan. Ohmsha Press.

Pazzani M. and Kibler D. (1991). The utility of knowledge in inductive learning. *Machine Learning, 9, 1,* 57-94.

Quinlan, R. 1990. Learning logical definitions from relations. *Machine Learning, 5, 3,* 239-266.

Spackman, K. (1988). Learning Categorical Decision Criteria in Biomedical Domains. In *Proceedings of the Fifth International Conference on Machine Learning* . Ann Arbor, MI. Morgan Kaufmann.