HYDRA: A Noise-tolerant Relational Concept Learning Algorithm

Kamal M. Ali **Michael J. Pazzani**
Department of Information and Computer Science
University of California
Irvine, CA 92717
{ali, pazzani}@ics.uci.edu

Abstract

Many learning algorithms form concept descriptions composed of clauses, each of which covers some proportion of the positive training data and a small to zero proportion of the negative training data. This paper presents a method using likelihood ratios attached to clauses to classify test examples. One concept description is learned for each class. Each concept description competes to classify the test example using the likelihood ratios assigned to clauses of that concept description. By testing on several artificial and "real world" domains, we demonstrate that attaching weights and allowing concept descriptions to compete to classify examples reduces an algorithm's susceptibility to noise.

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 classification error on an independent test set can be attributed to disjuncts[1] that cover a small number of positive examples. On noisy data, DNF concept learners typically learn a few reliable disjuncts and many unreliable disjuncts (small disjuncts) each of which covers a small number of positive training examples.

This paper proposes an approach based on learning one probabilistic concept description per class and allowing such concept descriptions to compete to classify test examples. We also present a method for estimating the reliability of the learned Horn clauses that is based on our experience in reducing the effect of small disjuncts (Ali and Pazzani, forthcoming). This gives clauses that cover few positive examples (usually the unreliable clauses) smaller weights in classifying test examples. HYDRA learns clauses in a manner similar to FOIL (Quinlan, 1990) and FOCL (Pazzani & Kibler, 1992). It then estimates the reliability of each clause and uses this information to increase its tolerance to noise. In the following section, we briefly explain how FOIL learns. Section 2 then presents HYDRA and the semantics associated with our formulation of clause reliability. Finally, in Section 3 we present the results of our experiments and compare HYDRA to related work.

1. We will refer to clauses and disjuncts interchangeably.

1.1 FOIL

FOIL builds a concept description for the target relation in terms of a conjunction of Horn clauses[2] over a pre-supplied set of "background" relations. The input to FOIL consists of positive and negative examples of the target relation and full extensional definitions of background relations. FOIL learns clauses one at a time. FOIL starts to learn a clause body by finding the literal with the maximum information gain, and continues to add literals to the clause body until the clause does not cover any negative examples. After learning a clause, FOIL removes the positive examples covered by that clause from further consideration. FOIL continues to build new clauses until each positive example has been covered.

1.2 Previous work

On noisy data sets, building clauses in this manner over-fits the training data leading to concept descriptions with high error rates. Previous work on learning first-order concept descriptions from noisy data has concentrated on using MDL approaches (Rissanen, 1978) to pre-prune the concept being learned (Quinlan's (1990) MDL algorithm for FOIL and Muggleton *et al.'s* (1992) HP-compression algorithm for GOLEM). Other work has concentrated on learning discriminant descriptions with certainty-factor attachment (Bergadano, *et al.*, 1988), using significance tests (Dzeroski and Bratko, 1992) or cross-validation (Brunk and Pazzani, 1991). In this paper we will compare HYDRA to Brunk and Pazzani's application of Reduced-Error Pruning (REP, Quinlan, 1987) to FOIL because they have shown REP to be more effective than Quinlan's MDL based approach when learning some relational concepts corrupted by noise.

1.3 Reduced-Error Pruning (REP)

REP splits the training data into two sets which we will call the learning data and the pruning data. FOIL is used to create a concept description using only the learning data. REP takes the concept description and applies syntactic operators to make small transformations to the concept description. It measures the merit of each operator application by comparing the numbers of correct

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

classifications (on the pruning data) before and after application of the operator. The transformations resulting from the best operator application are kept and the process is repeated until the application of any operator would result in a decrease in accuracy. Brunk and Pazzani used literal deletion and clause deletion as operators in their work.

2. HYDRA

HYDRA was developed to learn concept descriptions that reduce the small disjuncts problem by attaching weights to clauses. HYDRA also reduces errors of commission by building concept descriptions for each class and allowing them to compete to classify test examples. Finally, HYDRA uses a literal search metric that trades off coverage against fit more highly in favor of coverage. Thus HYDRA builds fewer, more reliable clauses than FOIL on noisy data.

2.1 Knowledge Representation and Classification

The method of learning probabilistic relational concept descriptions will be presented in Section 2.2. Here, we discuss how such descriptions are used to represent concepts and used for classification. HYDRA forms a concept description for each class. Each clause of each concept description has an associated likelihood ratio with values in $[0,\infty)$. One such annotated clause is shown below:

$$a(X,Z) \wedge a(Z,X) \rightarrow Class_i(X,Z)\ [LS = 2.3]$$

Letting τ denote an arbitrary example, the classificatory reliability of the *j-th* clause of the *i-th* class is estimated using the following ratio of probabilities (LS value, Duda *et al.,* 1979).

$$ls_{ij} = \frac{p(clause_{ij}(\tau)=true | \tau \in Class_i)}{p(clause_{ij}(\tau)=true | \tau \notin Class_i)}$$

We have chosen this heuristic rather than a measure of how sufficient the clause is for the class because there have been indications that the accuracy of some clauses (as measured on the training set) may not be good indicators of their accuracies in general (Muggleton *et al*., 1992). Furthermore, given that a learning algorithm aims to find clauses that are highly sufficient (accurate) for the class, our experiments and those in Muggleton *et al*. (1992) indicate that coverage of positive training examples is a good measure of the relative reliability (significance) among such clauses. However, positive coverage does not take into account any negative training examples that the clause may cover. The LS measure does take this into account by dividing the proportion of positive training examples covered by the clause by the proportion of negative training examples covered by the clause.

Classification of an example proceeds by considering only clauses that were satisfied by the test example. Among these clauses, the clause with the highest measure of reliability (LS) is deemed the winner. The test example is attributed to the class associated with the winning clause (ties are broken randomly). When a test example satisfies no clause of any class, HYDRA guesses the most frequent class.

2.2 Learning in HYDRA

HYDRA differs from FOIL in three major ways. First, HYDRA learns a concept description for each class so that each description can compete to classify a test example. This is accomplished by treating the training examples of each class in turn as positive training examples and the examples of all other classes as negative examples.

Second, after all the clauses have been learned, HYDRA forms an estimate of the logical sufficiency odds multiplier ls_{ij} associated with each clause. The entire training data is used when estimating ls_{ij}. Because some clauses may only cover a few examples, we have to address the problem of estimating probabilities from small numbers of examples. Considerable research in this area (Cestnik, 1990; Smyth *et al*., 1992) has indicated that in order to estimate a probability from small numbers of examples, the Laplace estimate and the m-estimate are preferable to a straightforward ratio of frequencies.

We use Laplace's law of succession to compute estimates for the two conditional probabilities in ls_{ij}. 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 for $P(X=v)$ is $(n_i+1)/(N+2)$. In order to estimate the numerator of ls_{ij} we note that the positive examples can be split into two classes: those that satisfy the clause and those that do not. Similarly using the Laplace estimate for the denominator gives us:

$$ls_{ij} = ls(p,n,p_0,n_0) = \frac{(p+1)/(p_0+2)}{(n+1)/(n_0+2)}$$

where p_0 and n_0 respectively denote the numbers of positive and negative examples (of $Class_i$) in the training data and p and n denote the numbers of examples covered by the clause.

The third difference between HYDRA and FOIL is that HYDRA uses a hill-climbing metric, *ls-conten*t, that is used to select which literal to conjoin to the clause currently being learned. Consider HYDRA as it chooses among literals to add to the *j-th* clause of class *i*. Let p and n denote the numbers of positive and negative examples not covered by by the *j-1* previously learned clauses. Assume that the *j-th* clause already consists of k literals, the conjunction of which covers p_k positive and n_k negative examples. Let p_{k+1} and n_{k+1} denote the numbers of positive and negative examples that would be covered if the literal under consideration were to be conjoined to the other literals of the *j-th* clause. Then the gain in ls-content is

$$ls(p_{k+1},n_{k+1},p,n)\times p_{k+1} - ls(p_k,n_k,p,n)\times p_k$$

where ls-content is the product of LS and p. If there are no

literals that cause an increase in ls-content or if the clause no longer covers any negative examples, HYDRA completes the clause, otherwise it conjoins the best candidate literal and the current clause. HYDRA performs better using ls-content than it does using information gain (see section 3.6).

3. Experimental Results

In this section, we show that the three major changes we have made to transform FOIL into HYDRA significantly reduce classification error rates in noisy domains although they slightly increase error rates when learning a concept best expressed in necessary and sufficient form. Our experiments compare HYDRA, FOIL and two noise-tolerant relational learners: REP (applied to FOIL) and mFOIL (Dzeroski and Bratko, 1992). The experiments were done on two artificial domains: the King-Rook-King chess end-game domain (KRK, Muggleton *et al.,* 1989) and the King-Rook King-Pawn domain (KPa7KR, Holte *et al.,* 1989). We also used the natural data sets of breast cancer recurrence, lymphography and E-Coli DNA promoters. Finally, in this section, we also do a step by step analysis of how FOIL is converted to HYDRA.

3.1 Description of the domains

In the KRK domain the goal is to differentiate *illegal* and *legal* board positions where a chess board is classified as *illegal* if either king is in check or if more than one piece occupies the same square. For all the natural data sets, HYDRA forms concept descriptions containing constants (Silverstein & Pazzani, 1991) and the relation *equal*. This allows attribute-attribute comparisons in addition to attribute-value comparisons. We converted the data sets which are represented as attribute vectors into tuples of the same length as the attribute vectors. For domains where some attributes were ordered, we included the relations $<$, $>$, $\leq$ and $\geq$. Missing values were represented by a special value, *unknown*. The only domain with domain-specific relations was the DNA domain. The relations used for this domain were *equal* and *equal-family* (the four bases {A,G,T,C} are split into two families: {A,G} and {T,C}).

3.2 Experimental methodology

We trained all of the algorithms on identical randomly selected training data and tested them on identical randomly selected testing data. This process was repeated twenty times. For the artificial domains where we could generate examples (KRK) or where we could draw examples from a large data set (KPa7KR) we used 1000 testing examples. For domains where the data was more limited, we trained on two-thirds of the data and tested on the remaining one-third. For the promoters domain, we used leave-one-out testing to allow comparability with other work on this domain. In all of the experiments in this section the expression "significantly better" means better as indicated by a unpaired two-tailed t-test with a confidence level of 0.95. A 20% level of class noise means the probability of randomly reassigning the class label is 0.2. A 5% level of tuple (attribute) noise means the probability of randomly reassigning each attribute is 0.05.

3.3 Comparison of HYDRA and FOIL

Table 1 and Figure 1 compare FOIL's accuracy rate to that of HYDRA. The natural data sets are considered noisy as their target concepts may not be expressible in DNF form using the available attributes or relations. FOIL is able to attain lower error rates only for noise-free data of the concept *illegal* from the KRK domain because that concept is easily expressible using first-order Horn clauses. On the noise-free KRK data set, FOIL is assured of an accuracy equal to the base rate of ~70% even if it builds a null concept description. FOIL and FOIL using REP only build concept descriptions for the positive class (*illegal* in the case of the KRK domain)[3]. As the class *legal* occurs with frequency approximately equal to 0.7, FOIL and REP can attain ~70% accuracy by not building any clauses. HYDRA is significantly more accurate than REP on noise-free data because REP is tailored to do well on noisy domains by assuming errors due to noise cannot be correlated between the learning and the pruning sets. For small, noise-free data sets, REP prunes significant clauses because such a clause may not cover any examples in the small pruning set. On noise-free data, HYDRA learns clauses that do not cover negative training examples thus approximating a DNF concept description. This can be seen in Table 1 by

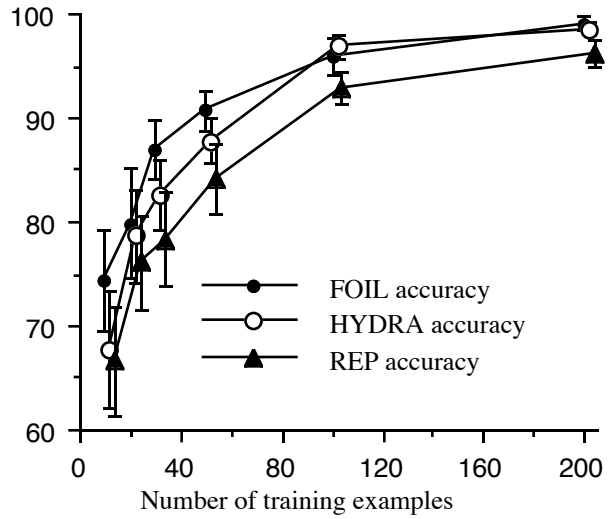

Figure 1. Comparison of HYDRA, REP and FOIL on noise-free examples from the KRK domain. The height of each vertical bar is equal to two standard deviations.

3. Experiments we conducted showed that concept descriptions built by FOIL for *legal* are less accurate than those for *illegal* because the background relations are tailored to learn *illegal*.

comparing the numbers of clauses learned by FOIL to the numbers of clauses learned by HYDRA for class *illegal*. Table 1 also compares HYDRA and FOIL on noisy training data from the KRK domain - using noise free test data. For noisy data from the KRK domain, FOIL's accuracy quickly climbs to ~82% as the number of training examples are increased but fails to increase substantially above that figure. The number of clauses built increases linearly with the number of noisy training examples, indicating FOIL is over-fitting the data. HYDRA fits the data more than REP but less than FOIL.

Domain or # of eg.s	Accuracy by FOIL	# of clauses	Accuracy by HYDRA	# of clauses
KRK data sets without noise				
10	**74.4**	2.0	67.7	(2.0,1.3)
20	79.8	2.8	78.6	(3.0,2.5)
30	**86.9**	3.8	82.5	(3.8,3.2)
50	**90.7**	5.0	87.8	(4.9,4.4)
100	95.9	6.1	96.8	(6.2,5.8)
200	99.1	7.3	98.5	(7.3,7.5)
KRK data sets with 20% Class noise				
10	68.8	2.0	68.9	(2.1,1.8)
20	74.4	3.7	71.9	(3.6,3.1)
30	78.4	4.9	78.7	(4.6,4.0)
50	81.3	7.4	81.0	(6.8,5.8)
80	82.7	11.0	**85.8**	(9.1,7.4)
160	83.7	21.1	**90.6**	(8.8,8.1)
320	82.4	38.7	**93.8**	(11.4,9.9)
480	83.1	54.3	**95.6**	(11.5,13.4)
Natural data sets				
Cancer	63.5	29.5	66.6	(5.3,7.8)
Lymph.	79.4	(1.1,8.0, 8.4,1.0)	81.4	(1.6,8.4, 7.6,1.0)
KPa7KR	90.3	7.6	**94.7**	(7.6,8.3)
Promoters	73.6	4.0	76.4	(5.0,6.1)

Table 1. Comparison of "vanilla" FOIL and HYDRA. Values in bold type indicate a statistically significant accuracy difference. Entries in the last column are vectors because HYDRA builds a set of clauses for each class. A variant of FOIL (algorithm 2 in section 3.6) was run instead of FOIL on the lymphography data because that data set contains more than two classes which "vanilla" FOIL cannot handle.

Experiments we conducted with other noise levels for the KRK domain reaffirm the basic result that HYDRA performs significantly better than FOIL on noisy data.

Figure 2 illustrates one reason for HYDRA's accuracy on noisy data sets: its ability to reduce the small disjuncts problem. In a system without such a problem, all disjuncts would be equally accurate so one would expect that the disjuncts that cover a given percentage of the examples would be responsible for that same percentage of the errors. Because the curves for FOIL and HYDRA are above the diagonal however, they indicate that clauses that match small proportions of the test examples contribute

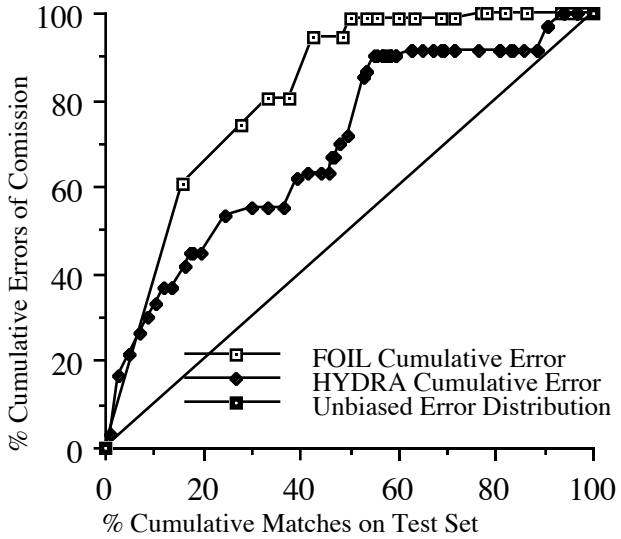

Figure 2. Clauses matching a small proportion of the test set contribute disproportionately to errors of commission. Both algorithms were trained on 160 examples from the KRK domain with 5% tuple noise.

disproportionately to errors of commission. However, this problem is less severe for HYDRA because it assigns smaller weights to clauses that cover few examples and are likely to be unreliable. The line for HYDRA in Figure 2 corresponds to clauses learned for both classes. In order to produce Figure 2, for each clause, we kept track of the number of test examples it matched and the number of errors it made. This yielded a sequence *S* of triples <clause,matches,errors> which were sorted in ascending order by matches. Then we used this code to make Figure 2:

```
current_matches = matches of head element of S
for <clause,matches,errors> in S do begin
   if matches ≠ current_matches begin
      plot( Cumulative_matches / Total_matches,
            Cumulative_errors / Total_errors)
      current_matches = matches
   end
   Cumulative_matches = Cumulative_matches + matches
   Cumulative_errors = Cumulative_errors + errors
end
plot( Cumulative_matches / Total_matches,
      Cumulative_errors / Total_errors)
```

3.4 Comparison of HYDRA and REP

Brunk and Pazzani show that on noisy data sets from the KRK domain, REP performs better than FOIL and Quinlan's minimum encoding length algorithm. Accordingly, this section compares HYDRA to REP and shows that using weights performs better than REP (on the domains we tested). Table 2 compares HYDRA and REP. HYDRA is significantly more accurate than REP on noisy data from the KRK domain when there are a limited number

Domain	HYDRA Accuracy	HYDRA # of clauses	REP Acc.	REP # of clauses
With 20% class noise				
KRK10	**68.9**	(2.1,1.8)	63.0	0.3
KRK20	71.9	(3.6,3.1)	69.6	0.9
KRK30	**78.7**	(4.6,4.0)	75.4	1.2
KRK50	**81.0**	(6.8,5.8)	78.0	2.2
KRK80	**85.8**	(9.1,7.4)	82.2	2.8
KRK160	90.6	(8.8,8.1)	91.2	4.3
KRK320	93.8	(11.4,9.9)	92.7	6.5
KRK480	95.6	(11.5,13.4)	96.3	6.9
With 5% tuple noise				
KRK10	71.3	(1.9,1.7)	67.5	0.4
KRK20	**78.1**	(3.1,2.6)	74.3	1.0
KRK30	**83.6**	(4.3,3.9)	74.9	1.1
KRK50	**86.8**	(5.5,5.2)	81.7	1.9
KRK80	**91.5**	(8.0,6.7)	85.5	2.9
KRK160	93.3	(10.1,8.9)	91.7	3.9
KRK320	96.1	(13.3,11.5)	95.7	5.7
KRK480	97.5	(17.6,16.3)	97.1	7.0
Natural data sets				
cancer	66.6	(5.3,7.8)	67.9	3.4
lymph.	**81.4**	(1.6,8.4, 7.6,1.0)	75.8	(0.4,2.5 2.6,0.2)
KPa7KR	**94.7**	(7.6,8.3)	91.1	2.9
Promoters	76.4	(5.0,6.1)	78.0	1.9

Table 2. Comparison of HYDRA and REP. Accuracy figures in bold indicate statistically significant advantage over the other algorithm.

of training examples because the pruning set must exceed a critical size for REP to be effective. Table 2 illustrates that for small numbers of examples from the KRK domain, REP prunes away almost all the concept description (column 5) so its reasonable accuracy is a result of the fact that even the null concept description attains ~70% accuracy in this domain (REP is highly accurate on test examples of *legal* and highly inaccurate on test examples of *illegal)*. Asymptotically, both algorithms approach 100% accuracy on the noisy data sets from the KRK domain. However, in the range where both algorithms are approximately equal in accuracy, REP is much more expensive in terms of learning cost. This is substantiated by Cohen (1993) who demonstrates that REP has an asymptotic time complexity of $\Omega(n^4)$. HYDRA by comparison, only needs to evaluate each learned clause over the training set once, in order to estimate the LS values.

3.5 Comparison of HYDRA and mFOIL

mFOIL (Dzeroski and Bratko, 1992) is a noise-tolerant relational learning algorithm, based on FOIL. mFOIL differs from FOIL in using beam search with a beam width of 5 and it can use either the Laplace or the m-estimate (Cestnik, 1990) of expected accuracy in place of information gain.Let p and n respectively denote the numbers of positive and negative examples covered by a literal, m denote a parameter to the system and p^+ denote the prior of the positive class, then the Laplace estimate is $(p+1)/(p+n+2)$ and the m-estimate is $(p+mp^+)/(p+n+m)$. Table 3 presents an empirical comparison of HYDRA and mFOIL. HYDRA does significantly better than mFOIL using Laplace. For mFOIL using m = 0.01 (empirically determined to be the best value for the KRK domain by Dzeroski and Bratko, 1992), HYDRA does approximately as well as mFOIL for class noise and does a little better for tuple noise.

3.6 Analysis of the progression from FOIL to HYDRA

In this section, we do a step by step analysis of a progression from FOIL to HYDRA. There are three major differences between FOIL and HYDRA:

- FOIL learns Horn clauses for positive examples of a single class. HYDRA learns a set of Horn clauses for each class.
- FOIL uses information gain to select literals while HYDRA uses ls-content.
- HYDRA associates weights (LS values) with each clause.

Figure 3 shows the comparisons that we will make. For each comparison, the impact of only one difference will be assessed. In the first three comparisons, we present a

Level of noise	mFOIL using Laplace	mFOIL using m = 0.01	HYDRA accuracy	HYDRA # of clauses
Class noise				
0%	95.1	95.6	96.8	(6.2,5.8)
5%	92.7	94.3	91.9	(8.0,6.8)
10%	88.9	92.0	91.0	(9.3,7.9)
15%	86.4	90.0	88.9	(9.9,7.8)
20%	84.6	88.1	88.9	(10.3,8.4)
Tuple noise				
5%	90.1	91.9	90.0	(9.1,7.9)
10%	80.4	84.6	86.8	(9.9,7.5)
15%	78.3	80.1	83.0	(10.2,8.0)
20%	73.7	75.0	79.7	(10.2,8.3)

Table 3. Comparison of mFOIL to HYDRA on the KRK domain. All results are for 100 examples from the KRK domain. Accuracies for mFOIL represent averages over 5 trials.

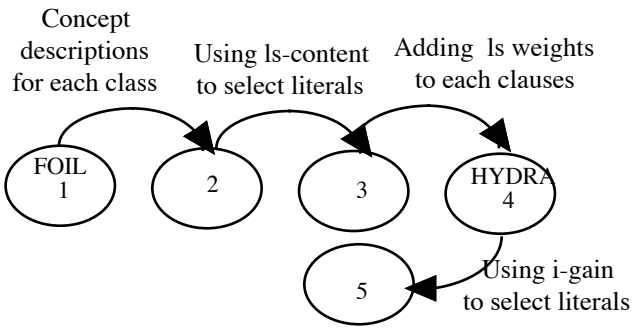

Figure 3. A progression of changes that converts FOIL into HYDRA, (followed by algorithm 5, that differs from HYDRA only in the use of information gain to select literals).

progression of configurations from FOIL to HYDRA that allows one to see what impact each change had in the light of preceding changes.

Note that the comparison between algorithm 3 and HYDRA (algorithm 4) is a "lesion" study, because exactly one component of HYDRA (the conflict resolution method used to determine the class when clauses from more than one class are satisfied) is changed to form algorithm 3. Similarly, the comparison between HYDRA and algorithm 5 is a lesion study in which the literal selection metric) is changed. It does not make sense to consider the third lesion (not learning a description for each class) because this change alone is not meaningful since it also eliminates the need for a conflict resolution strategy.

First, we compare FOIL (algorithm 1 in Figure 3) to a version of FOIL (algorithm 2 in Figure 3) that only differs from FOIL in that it learns a concept description for each class. When classifying an example, if an example satisfies a clause of more than one concept description, algorithm 2 needs to estimate the reliability of the contending clauses in order to resolve this conflict. HYDRA resolves such conflicts by selecting the class with the highest LS value. Algorithm 2 resolves conflicts by selecting the class corresponding to the clause that covers the greater number of positive training examples, breaking ties randomly. When a test example satisfies no clause of any class, algorithm 2 guesses the most frequent class.

Domain	**FOIL**	**2**	**3**	**HYDRA**	**5**
Noise-free:					
KRK100	97.0	95.9	95.0	**96.8**	97.1
KRK200	99.1	**98.2**	97.9	98.5	98.6
With 20% class noise					
KRK160	83.9	**87.3**	85.9	**90.6**	**88.0**
KRK320	83.8	**90.9**	**81.5**	**93.8**	**91.4**
With 5% tuple noise					
KRK160	90.6	91.0	89.6	**93.3**	92.4
KRK320	90.7	**94.0**	**89.0**	**96.1**	94.9
Natural data sets					
Cancer	63.5	**66.4**	**62.5**	**66.6**	**63.8**
Lymph.	-	79.4	80.5	81.4	78.8
Kpa7KR	90.3	**94.1**	94.7	94.7	94.2
Promoters	73.6	61.3	76.4	76.4	70.8

Table 4. Progression from FOIL to HYDRA. Figures in bold indicate statistically significant differences as measured with respect to the column immediately to the left. All figures represent accuracies averaged over 20 trials.

As Table 4 indicates, algorithm 2 is significantly more accurate than FOIL on noisy data. It makes fewer errors of commission than FOIL. Consider the KRK domain. FOIL only learns a definition for the concept *illegal* so an error of commission occurs when it classifies an example of class *legal* as belonging to *illegal*. Algorithm 2 may not make such an error in the same situation if that example also satisfies a clause of *legal* that is more reliable than the contending clause from *illegal*. Thus, algorithm 2's conflict resolution strategy would correctly award that example to *legal*. For the noisy KRK, cancer and lymphography data sets, such conflicts occur on approximately 20% of the test examples. On approximately 1-2% of the test examples in these domains, it has to guess the most frequent class. However, in the promoters domain algorithm 2 has to guess the most frequent class on about 30% of the test examples. Since both classes are approximately equally frequent in this data set, algorithm 2 concedes errors on 15% of the test data from just this source of error. This explains why algorithm 2 has a low accuracy rate (61.3%) (Table 4) on the promoters data set. The promoters concept is hard to learn for both algorithm 2 and HYDRA because it is highly disjunctive with respect to the available relations. In summary, learning a concept description for each class helps on all the noisy data sets except for the highly disjunctive DNA promoters data set.

Now we consider the effect of replacing information gain by ls-content and learning a concept description for each class (The transition from algorithm 2 to algorithm 3 in Figure 3). Algorithm 3 leads to a slight (although not significant) or a significant ($p < 0.05$) decrease in accuracy on all the variants of the KRK domain and the cancer domain. Interestingly, ls-content helps on the highly disjunctive DNA and the lymphography data sets. On these sets, we observed that when using information gain, many test examples were uncovered by any clause of any concept description. On such occasions, algorithm 2 has to guess the most frequent class. Algorithm 3, using ls-content builds clauses with greater coverage so its has to guess less frequently and consequently has a higher accuracy on these data sets.

Next, we consider the impact of adding LS weights to the concept description learned by algorithm 3. This is algorithm 4, which we call HYDRA. Table 4 shows that adding weights helps for the KRK and some other domains and does not hurt in other cases. This increase in accuracy is due not to building a different concept description (HYDRA uses the same concept description as algorithm 3). Rather, the increased accuracy is due to weighing clauses according to a measure of their reliability. Note that the only source that can cause differences in accuracies between algorithm 3 and HYDRA is the conflict resolution mechanism. Algorithm 3 resolves conflicts by using positive coverage and HYDRA resolves conflicts using the product of LS and prior odds. These two methods only differ when comparing clauses covering negative examples in the training data. Thus, Table 4 shows that HYDRA is more accurate than algorithm 3 on noisy data sets from the KRK domain because both algorithms learned clauses covering negative examples in that domain. This also occurs for the cancer

domain. By contrast, both algorithms do not learn many clauses covering negative examples in the DNA and lymphography data sets. Thus, changing the conflict resolution mechanism by adding weights causes little or no change in accuracy.

Finally, we consider the impact of changing the literal selection metric from ls-content in HYDRA to information gain, creating algorithm 5. HYDRA using ls-content is often significantly more accurate than algorithm 5. On none of the domains we used, was HYDRA significantly less accurate than algorithm 5. On noisy data, the average accuracy of HYDRA was always higher than that of algorithm 5. The comparison of HYDRA and algorithm 5 indicates that the system of weighing reliability must be supported by clauses that do not over-fit the data. Another reason algorithm 5 is significantly less accurate than HYDRA is that it uses the same concept description that algorithm 2 uses, hence it too makes many errors of omission through frequently having to guess the most frequent class.

4. Conclusions and Future Work

We have presented a method using likelihood multipliers that demonstrably reduces the small disjuncts problem and thereby increases predictive accuracy on noisy domains. This method has been tested on relational and attribute-value domains. HYDRA demonstrates the utility of learning a concept description for each class that compete to classify examples, and the utility of estimating the reliability of clauses. This method does better than REP which has been the most accurate algorithm to date on noisy relational data such as the KRK data.

We plan to extend HYDRA to build several independent concept descriptions per class and then combine evidence from these models (averaging multiple models, Buntine, 1991). We feel that learning multiple models will help HYDRA to further reduce the problems that hill-climbing systems like FOCL experience in noisy domains.

Acknowledgements

This research was supported in part by grant F49620-92-J-0430 from Air Force Office of Scientific Research. We would like to like to acknowledge Matjaz Zwitter and Milan Soklic of the Institute of Oncology at the University Medical Center, Ljubljana, Yugoslavia for the breast cancer recurrence data. Thanks also to William Cohen, Michael Cameron-Jones, Dennis Kibler, Donato Malerba and Ross Quinlan for reviews of this paper, to Cliff Brunk for suggestions on HYDRA and to the IJCAI reviewers.

References

Ali K. and Pazzani M. (forthcoming). Reducing the small disjuncts problem by learning probabilistic concept descriptions. In Petsche T., Judd S. and Hanson S. (ed.s), *Computational Learning Theory and Natural Learning Systems, Vol. 3*. Cambridge, Massachusetts. MIT Press.

Bergadano F., Giordana A. and Saitta L. (1988). Automated concept acquisition in noisy environments. In *IEEE Pattern Analysis and Machine Intelligence, 10*.

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

Cestnik B. (1990). Estimating probabilities: A crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence*. Stockholm, Sweden. Pitman Press.

Cohen W. Efficient Pruning Methods for Separate-and-Conquer Rule Learning Systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. Chambery, France. 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, Scotland. 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.

Muggleton S., Bain M., Hayes-Michie J. and Michie D. (1989). An experimental comparison of human and machine-learning formalisms. *Sixth International Workshop on Machine Learning*. Ithaca, NY. Morgan Kaufmann.

Muggleton S., Srinivasan A. and Bain M. (1992). Compression, Significance and Accuracy. In *Proceedings of the Ninth International Workshop (ML92)*. Aberdeen, Scotland. Morgan Kaufmann.

Pazzani M. and Kibler D. (1992). The utility of knowledge in inductive learning. *Machine Learning, 9, 1,* 57-94.

Rissanen J. (1978). Modeling by Shortest Data Description. *Automatica, 14*.

Quinlan R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27,* 221-234.

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

Silverstein G. and Pazzani M. (1991). Relational cliches: Constraining constructive induction during relational learning. In *Proceedings of the Eighth International Workshop on Machine Learning*. Evanston, IL. Morgan Kaufmann.

Smyth P., Goodman R. (1992). Rule Induction Using Information Theory. In G. Piatetsky-Shapiro & W. Frawley (Eds.) *Knowledge Discovery in Databases*. MIT Press.

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.