HYDRA: A Noise-tolerant Relational Concept Learning Algorithm

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

Technical Report 92-85

December 17, 1992

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

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 for attaching likelihood ratios to clauses and a method for using such ratios to classify test examples. This paper presents the relational concept learner HYDRA that learns a concept description 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.

Area: Learning (Induction from noisy data; First order Learning)

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 training set can be attributed to learned disjuncts[1] that cover a small number of positive examples. On noisy data sets, DNF concept learners typically learn a few reliable disjuncts (large disjuncts) and many more unreliable disjuncts, each of which covers a small number of positive examples (small disjuncts).

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) minimum encoding length algorithm for FOIL and Muggleton *et al.'s* (1992) HP-compression algorithm for GOLEM). Other work has concentrated on over-fitting noisy data but using flexible matching to tolerate noise (the POSEIDON system by Bergadano *et al.*, 1992), learning discriminant descriptions with certainty-factor attachment (Bergadano, Giordana and Saitta, 1988), using significance tests (Dzeroski and Bratko, 1992) or cross-validation (Brunk and Pazzani, 1991). 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 learned clauses that is based on our experience in reducing the effect of small disjuncts (Ali and Pazzani, 1993).

This paper presents the system HYDRA that builds concept descriptions consisting of Horn clauses with associated likelihood ratios. We empirically show that such concept descriptions have low error rates and reduce the effect of small disjuncts on noisy relational and attribute-value data.

HYDRA builds probabilistic relational clauses by attaching likelihood ratios (ratios of

1. We will refer to clauses and disjuncts interchangeably.

probabilities) to each clause. 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 sections, we briefly explain how FOIL learns. We will also review Reduced-Error Pruning (REP, Quinlan, 1987), a post-processing method designed to deal with noise (implemented for FOIL by Brunk and Pazzani, 1990). Section 2 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.

1.1 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 or until there is no literal with positive information gain. After learning a clause, FOIL removes the positive examples covered by that clause from further consideration. FOIL continues to build new clauses in this manner until each positive example has been covered by at least one clause.

On noisy data sets, over-fitting the training data leads to concept descriptions with high error rates. Quinlan (1990) introduced a noise-tolerant extension to FOIL that compared the length of an encoding of a clause to an encoding of the training examples covered by that clause. This algorithm stops conjoining literals to the clause if the encoding length of the clause exceeds the encoding length of the data. However, Brunk and Paz-

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 $(a \wedge b) \vee (d \wedge e)$.

zani (1991) experimentally showed that the encoding length approach applied to FOIL is ineffective because it results in less accurate concepts on noisy data sets. They also show that REP (as applied to FOIL) yields more accurate concept descriptions. Accordingly, in this paper we will compare HYDRA to REP.

1.2 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 operators to make small transformations to the concept description. It measures the merit of each operator application by comparing the numbers of correct 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 two operators in their work: delete the last literal of a clause *(delete-last-literal)* and delete an entire clause *(delete-clause)*.

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 metric that trades off coverage against fit more highly in favor of coverage. Thus HYDRA is a pre-pruning algorithm that builds fewer, more reliable clauses than FOIL.

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 $[0,\infty)$. A few such annotated clauses for some classes are shown below:

$$a(X,Z) \wedge a(Z,X) \rightarrow \mathrm{Class}_i(X,Z)\ [LS = 2.3]$$

$$a(X,Y) \wedge c(X,Y) \rightarrow \mathrm{Class}_i(Y,X)\ [LS=14.7]$$

$$d(X,Z,Z) \wedge b(Z,X) \rightarrow \mathrm{Class}_j(X,Z)\ [LS = 1.8]$$

To classify a new example τ, HYDRA ranks its estimates for $p(\tau \in \mathrm{Class}_i | CD_i(\tau)=\mathrm{true})$ (the probability that τ is an element of class_i given that it satisfies the *i-th* concept description) and attributes τ to the concept description (CD) that maximizes this posterior probability.

The following discussion explains how HYDRA computes $p(\tau \in \mathrm{Class}_i | CD_i(\tau)=\mathrm{true})$ for each class. $p(\tau \in \mathrm{Class}_i | CD_i(\tau)=\mathrm{true})$ can be rewritten as

$$p(\tau \in \mathrm{class}_i | CD_i(\tau)=\mathrm{true}) = p(\tau \in \mathrm{class}_i | (\mathrm{clause}_{i1}(\tau)=\mathrm{true}) \vee \cdots \vee (\mathrm{clause}_{in}(\tau)=\mathrm{true})) \quad [1]$$

FOIL and HYDRA aim to learn clauses that partition the positive examples so ideally there should only be one clause per concept description that is satisfied for a given τ. In practice, more than one clause (from one concept description) may be satisfied. In this case, we choose the clause that is most indicative of the hypothesis $\tau \in \mathrm{class}_i$:

$$p(\tau \in \mathrm{Class}_i | CD_i(\tau)=\mathrm{true}) \approx \mathrm{MAX}_{\mathrm{clause}_{ij} \in CD_i} [p(\tau \in \mathrm{Class}_i | \mathrm{clause}_{ij}(\tau)=\mathrm{true})] \quad [2]$$

where the maximum is taken over clauses of CD_i satisfied by τ. In order to calculate $p(\tau \in \mathrm{Class}_i | \mathrm{clause}_{ij}(\tau)=\mathrm{true})$ we use the notion of odds. Following Duda *et al.* (1979) we define the odds of a hypothesis H as odds(H) = p(H)/(1-p(H)). Recasting Bayes rule into odds form (Duda *et al.*, 1979), it then follows that:

$$\mathrm{odds}(\tau \in \mathrm{class}_i | \mathrm{clause}_{ij}(\tau)=\mathrm{true}) = \frac{p(\mathrm{clause}_{ij}(\tau)=\mathrm{true} | \tau \in \mathrm{Class}_i)}{p(\mathrm{clause}_{ij}(\tau)=\mathrm{true} | \tau \notin \mathrm{Class}_i)} \times \text{prior-odds}(\tau \in \mathrm{Class}_i) \quad [3]$$

Because odds and probabilities are positively monotonically related, the clause with the highest value for $p(\tau \in \mathrm{Class}_i | \mathrm{clause}_{ij}(\tau)=\mathrm{true})$ (our goal) is also the clause with the highest value for $\mathrm{odds}(\tau \in \mathrm{class}_i | \mathrm{clause}_{ij}(\tau)=\mathrm{true})$. HYDRA estimates all the quantities on the right hand side using frequencies over the training data[3]. In particular, the ratio on the right side is the weight (called LS) associated with clause_{ij} and is referred to as the sufficiency measure (Duda *et al.*, 1979). In summary, HYDRA uses these weights to assign τ to the class for which $p(\tau \in \mathrm{Class}_i | CD_i(\tau)=\mathrm{true})$ is maximal. When a test example satis-

3. Empirically, we find that if we estimate the prior odds $p(\tau \in \mathrm{Class}_i)/p(\tau \notin \mathrm{Class}_i)$ from the data, HYDRA performs worse than if we assign, uniform, uninformative priors to all the classes.

fies no clause of any concept description, HYDRA guesses the most frequent class.

Classification of an example proceeds as follows. For each clause that is true for the current test example HYDRA considers the product of the LS of that clause and the prior odds of the class to which that clause belongs. HYDRA attributes the test example to the class that had a clause with the maximal product[4]. The product of the LS and the prior odds represents the posterior odds of the test example belonging to that class (equation 3). As odds and probabilities are positively monotonically related, the class with the highest odds will be the class with the highest posterior probability. 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 according to the posterior probabilities $p(\tau \in \text{Class}_i | LCD_i{=}\text{true}, \tau)$. 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}. 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:

$$ls_{ij} = ls(p,n,p_0,n_0) = \frac{(p+1)/(p_0+2)}{(n+1)/(n_0+2)} \qquad [4]$$

where p_0 and n_0 respectively denote the numbers of positive and negative examples (of

4. Ties occur very infrequently and are broken randomly.

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 that is aimed at learning probabilistic concept descriptions. We call this metric ls-content and it is defined as follows. If a literal of the *j-th* clause covers *p* positive examples and *n* negative examples, and there were $p_{j,0}$ and $n_{j,0}$ examples uncovered after the first *j -1* clauses, its ls-content is defined as:

$$\text{ls-content}(p, n, p_{j,0}, n_{j,0}) = \text{ls}(p, n, p_{j,0}, n_{j,0}) \times p \qquad [5]$$

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 or if the clause no longer covers any negative examples, HYDRA completes the clause, otherwise it conjoins the literal and the current clause and resets *p* and *n* to reflect the numbers of examples that satisfy the clause with the new literal conjoined. 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 necessary and sufficient target concept. Our experiments compare HYDRA, FOIL and two noise-tolerant relational learners: REP (with 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, it is white's turn to move and there are three pieces: a white king, white rook and a black king. 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 four algorithms we compare here, the board was represented as a 6 tuple consisting of the file and rank coordinates of each piece (Pazzani and Kibler, 1991). In order to form a description for *illegal*($V_1..V_6$), HYDRA uses the relations *near(X,Y)* (X= Y+1 or X = Y-1), *between(X,Y,Z)* (X<Y<Z or X>Y>Z) and *equal*. For this domain, we only considered constant-free concept descriptions.

For all the natural data sets, HYDRA forms concept descriptions containing constants and includes the relation *equal* in addition to domain-specific relations. This allows attribute-attribute comparisons in addition to attribute-value comparisons (which have been proven to be useful in these domains). 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 <, >, ≤ and ≥ . 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 = 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 the identical randomly selected training data and tested them on the 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 paired two-tailed t-test with a confidence level of 0.95.

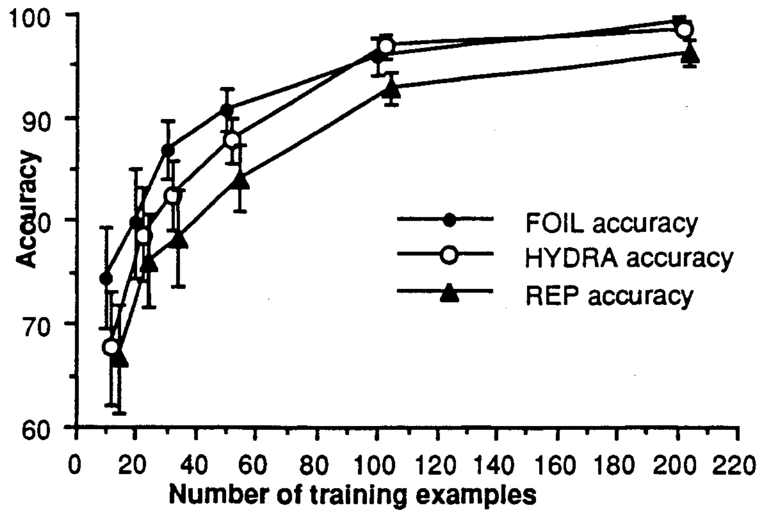


Figure 1. Comparison of HYDRA, REP and FOIL on noise-free examples from the KRK domain. The height of each vertical bar is equal to two standard deviations.

3.3 Comparison of HYDRA and FOIL

Table 1 and Figure 1 compare FOIL's accuracy rate to that of HYDRA, first on noise-free KRK examples and then on noisy data sets. 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* for the KRK domain)[5]. 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 examples and consequently have high LS values leading to a concept description that approximates a DNF form without weights. This can be seen in Table 1 by comparing the numbers of

5. 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*.

Domain	Number of training examples	FOIL accuracy	FOIL # of clauses	HYDRA accuracy	HYDRA # of clauses
Without noise					
KRK	10	**74.4** (9.8)	2.0	67.7 (11.1)	(2.0,1.3)
KRK	20	79.8 (10.6)	2.8	78.6 (9.0)	(3.0,2.5)
KRK	30	**86.9** (5.6)	3.8	82.5 (6.7)	(3.8,3.2)
KRK	50	**90.7** (3.9)	5.0	87.8 (4.5)	(4.9,4.4)
KRK	100	95.9 (3.6)	6.1	96.8 (2.2)	(6.2,5.8)
KRK	200	99.1 (1.2)	7.3	98.5 (1.3)	(7.3,7.5)
With 20% Class noise					
KRK	10	68.8 (11.1)	2.0	68.9 (14.6)	(2.1,1.8)
KRK	20	74.4 (9.9)	3.7	71.9 (11.2)	(3.6,3.1)
KRK	30	78.4 (8.3)	4.9	78.7 (7.1)	(4.6,4.0)
KRK	50	81.3 (6.9)	7.4	81.0 (6.7)	(6.8,5.8)
KRK	80	82.7 (5.8)	11.0	**85.8** (4.3)	(9.1,7.4)
KRK	160	83.7 (5.0)	21.1	**90.6** (3.5)	(8.8,8.1)
KRK	320	82.4 (2.5)	38.7	**93.8** (2.8)	(11.4,9.9)
KRK	480	83.1 (2.7)	54.3	**95.6** (2.6)	(11.5,13.4)
Natural data sets					
Cancer	190	63.5 (4.3)	29.5	66.6 (5.2)	(5.3,7.8)
Lymph.	99	79.4 (4.3)	(1.1,8.0,8.4,1.0)	81.4 (4.2)	(1.6,8.4,7.6,1.0)
KPa7KR	200	90.3 (2.5)	7.6	**94.7** (1.1)	(7.6,8.3)
Promoters	105	73.6	4.0	**76.4**	(5.0,6.1)

Table 1. Comparison of "vanilla" FOIL and HYDRA on various data sets. Values in bold type indicate a statistically significant accuracy difference. Standard deviation is indicated in parentheses for the accuracies columns. Standard deviation is not indicated for the promoters domain because leave-one-out testing was used on that domain. 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.

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[6]. 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. 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

6. For domains (such as KRK) where we could artificially generate examples, we tested on noise-free test data.

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. Because it builds multiple concept descriptions that may overlap in the instance space, it resolve conflicts in the overlapped region by examining posterior odds of the contending classes. 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.

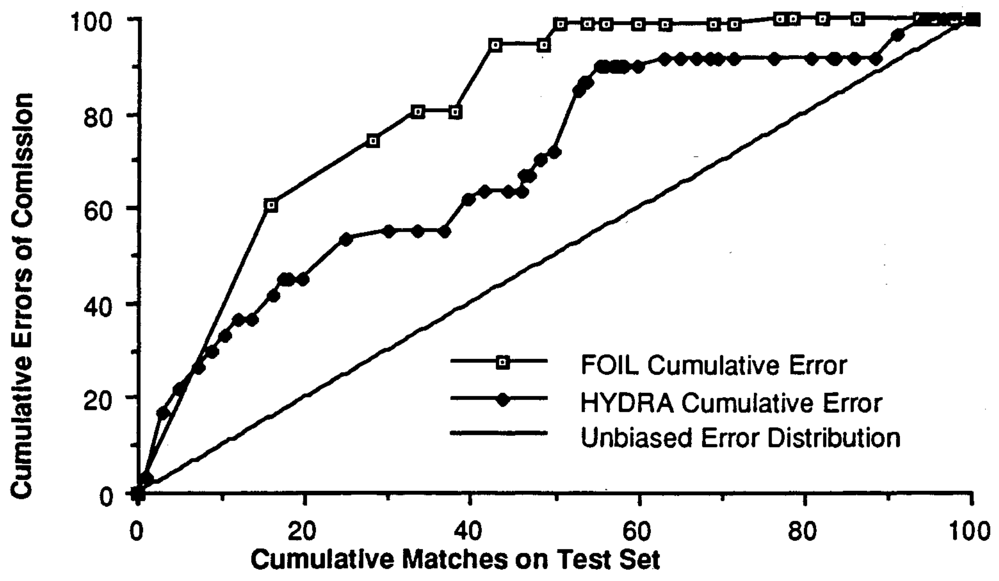
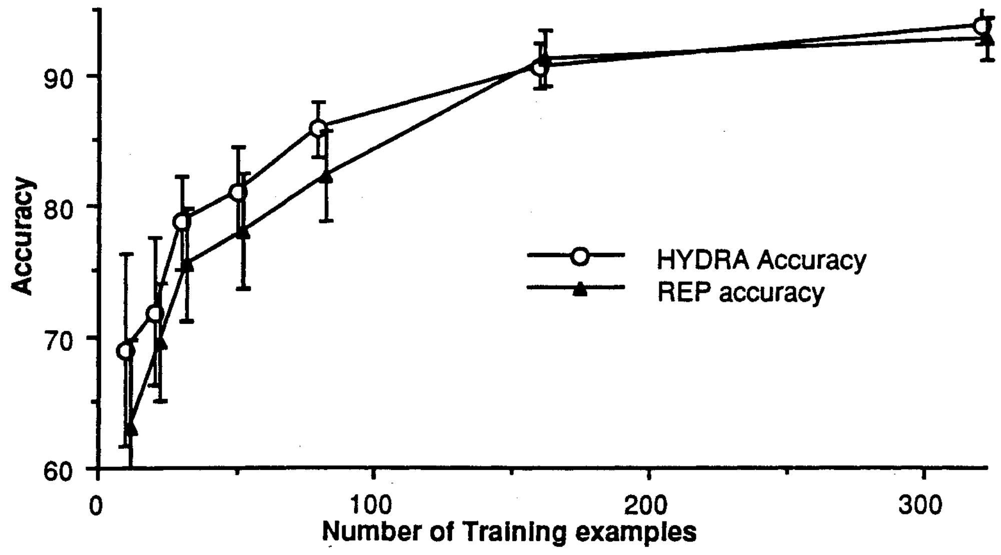
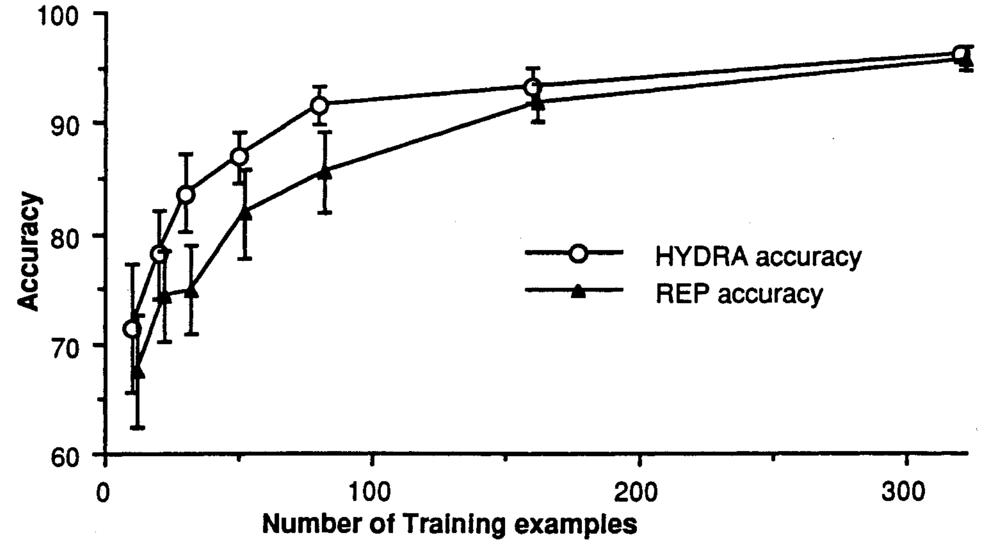


Figure 2. Clauses matching a small proportion of the test set contribute disproportionately to errors of commission. Both algorithms were trained on 160 examples from the KRK domain with 5% tuple noise.

Figure 2 illustrates one reason for HYDRA's accuracy on noisy data sets: its ability to reduce the small disjuncts problem. To produce this figure, we analyzed the output of FOIL and HYDRA as follows. The clauses learned were sorted according to the number of examples in the test data that satisfy the clause. Next, starting with clauses that cover the least number of examples, we added clauses to a set and recorded both the percentage of test examples that were matched by the clauses in the set, and the percentage of total errors that were committed by clauses in the set. Small disjuncts cause a problem in a system when disjuncts that cover a given percentage of examples are responsible for a greater percentage of the errors. 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, they indicate that clauses that match small proportions of the test examples contribute disproportionately to errors of commission. However, this problem is less severe in HYDRA than FOIL. For example, Figure 2 shows that clauses matching 40% or less of the test examples are responsible for about 80% of the errors in FOIL while they are responsible for about 50% of the error in HYDRA. This occurs because HYDRA assigns smaller weights to clauses that cover few examples and are likely to be unreliable.





Figures 3a and 3b. Comparison of HYDRA and REP on KRK with 20% class noise and 5% tuple noise, respectively. Standard-deviation bars are shown.

Number of eg.s	HYDRA Acc.	HYDRA Accuracy cpt.s	HYDRA # of clauses	HYDRA cpu sec.	REP Acc.	REP Acc. cpt.s	REP # of clauses	REP cpu sec.
With 20% class noise								
KRK10	**68.9** (14.6)	(55,76)	(2.1,1.8)	2.02	63.0 (13.4)	(22,83)	0.3	0.82
KRK20	71.9 (11.2)	(64,76)	(3.6,3.1)	5.98	69.6 (8.9)	(31,89)	0.9	2.98
KRK30	**78.7** (7.1)	(70,82)	(4.6,4.0)	9.24	75.4 (8.5)	(37,94)	1.2	5.98
KRK50	**81.0** (6.7)	(78,82)	(6.8,5.8)	26.7	78.0 (8.7)	(55,90)	2.2	20.9
KRK80	**85.8** (4.3)	(83,87)	(9.1,7.4)	53.4	82.2 (7.0)	(67,90)	2.8	67.9
KRK160	90.6 (3.5)	(84,94)	(8.8,8.1)	111	91.2 (4.4)	(80,97)	4.3	160
KRK320	93.8 (2.8)	(89,97)	(11.4,9.9)	252	92.7 (3.3)	(88,95)	6.5	1705
KRK480	95.6 (2.6)	(90,98)	(11.5,13.4)	340	96.3 (2.3)	(93,98)	6.9	7950
With 5% tuple noise								
KRK10	71.3 (11.6)	(55,79)	(1.9,1.7)	-	67.5 (10.0)	(18,93)	0.4	-
KRK20	**78.1** (7.9)	(62,86)	(3.1,2.6)	-	74.3 (8.1)	(32,95)	1.0	-
KRK30	**83.6** (7.1)	(75,88)	(4.3,3.9)	-	74.9 (7.9)	(34,96)	1.1	-
KRK50	**86.8** (4.8)	(77,92)	(5.5,5.2)	-	81.7 (7.9)	(56,95)	1.9	-
KRK80	**91.5** (3.4)	(86,94)	(8.0,6.7)	-	85.5 (7.3)	(70,93)	2.9	-
KRK160	93.3 (3.3)	(89,96)	(10.1,8.9)	-	91.7 (3.5)	(81,97)	3.9	-
KRK320	96.1 (1.2)	(92,98)	(13.3,11.5)	-	95.7 (2.1)	(90,98)	5.7	-
KRK480	97.5 (1.2)	(96,98)	(17.6,16.3)	-	97.1 (1.9)	(93,99)	7.0	-
Natural data sets								
cancer	66.6 (5.2)	(45,76)	(5.3,7.8)	-	67.9 (4.5)	(33,83)	3.4	-
lymph.	**81.4** (4.2)	(71,71, 89,100)	(1.6,8.4, 7.6,1.0)	-	75.8 (6.8)	(14,59, 94,0)	(0.4,2.5, 2.6,0.2)	-
KPa7KR	**94.7** (1.1)	(97,92)	(7.6,8.3)	-	91.1 (2.3)	(91,91)	2.9	-
Promoters	76.4 (42.7)	(83,70)	(5.0,6.1)	-	**78.0** (4.7)	(81,74)	1.9	-

Table 2. Comparison of HYDRA and REP. Figures in parentheses for accuracies columns indicate standard deviation. Accuracy components (cpt.s) indicate the breakdown of the overall accuracy by class. Accuracy figures in bold indicate statistically significant advantage over the other algorithm.

3.4 Comparison of HYDRA and REP

Brunk and Pazzani show that on noisy data sets from the 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 a pre-pruning method with weights performs better than REP (on the domains we tested) despite the problem of local information that besets pre-pruning methods. Figures 3a and 3b compare HYDRA and REP on noisy data from the KRK domain. HYDRA is significantly more accurate than REP on noisy data when there are a limited number 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 8) so its reasonable accuracy is a result of the fact thateven 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 approx-imately equal in accuracy, REP is much more expensive in terms of learning cost. Although Table 2 measures complexity in cpu seconds, the differences in complexity are large enough such that a first order approximation is sufficient here. This is substantiated by Cohen (1992) 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.

Level of noise	mFOIL using Laplace	mFOIL m = 0.01	HYDRA accuracy	HYDRA # clauses
Class noise				
0%	95.1	95.6	96.8 (2.2)	(6.2,5.8)
5%	92.7	94.3	91.9 (3.6)	(8.0,6.8)
10%	88.9	92.0	91.0 (4.4)	(9.3,7.9)
15%	86.4	90.0	88.9 (3.5)	(9.9,7.8)
20%	84.6	88.1	88.9 (3.1)	(10.3,8.4)
Tuple noise				
5%	90.1	91.9	90.0 (3.1)	(9.1,7.9)
10%	80.4	84.6	86.8 (3.7)	(9.9,7.5)
15%	78.3	80.1	83.0 (4.9)	(10.2,8.0)
20%	73.7	75.0	79.7 (5.1)	(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. Figures in parentheses for HYDRA accuracies are standard deviations. Accuracies for mFOIL represent averages over 5 trials. Dzeroski and Bratko did not give standard deviations.

3.5 Comparison of HYDRA and mFOIL

mFOIL (Dzeroski and Bratko, 1992) is a noise-tolerant relational learning algorithm, based on FOIL. mFOIL can use either the Laplace or the m estimate (Cestnik, 1990) of expected accuracy in place of information gain and it uses beam search with a beam width of 5. If p and n respectively denote the numbers of positive and negative examples covered by a literal, m denotes a parameter to the system and p^+ denotes 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 and 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 4 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 progression of configurations from FOIL to HYDRA that allows one to see what impact each change had in the light of preceding changes.

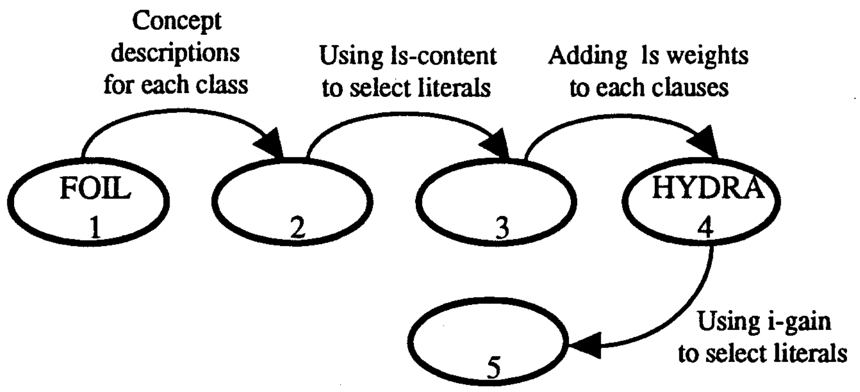

Concept descriptions for each class
Using ls-content to select literals
Adding ls weights to each clauses
FOIL 1
2
3
HYDRA 4
5
Using i-gain to select literals

Figure 4. 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)

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 another component of HYDRA (the literal selection metric) is changed.[7]

First, we compare FOIL (algorithm 1 in Figure 4) to a version of FOIL (algorithm 2 in Figure 4) 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 posterior odds. 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.

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

7. Note that although there are three differences between HYDRA an FOIL, it does not make sense to consider a third lesion (not learning a concept definition for each class), because this change alone is not meaningful (since it also eliminates the need for a conflict resolution strategy).

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.

Domain	**Number**	**FOIL**			**HYDRA**	
	of eg.s	**1**	**2**	**3**	**4**	**5**
Noise-free:						
KRK	100	97.0	95.9	95.0	**96.8**	97.1
KRK	200	99.1	**98.2**	97.9	98.5	98.6
With 20% class noise						
KRK	160	83.9	**87.3**	85.9	**90.6**	**88.0**
KRK	320	83.8	**90.9**	81.5	**93.8**	**91.4**
With 5% tuple noise						
KRK	160	90.6	91.0	89.6	**93.3**	92.4
KRK	320	90.7	**94.0**	89.0	**96.1**	94.9
Natural data sets						
Cancer	190	63.5	**66.4**	**62.5**	**66.6**	**63.8**
Lymph.	99	-	79.4	80.5	81.4	78.8
Kpa7KR	200	90.3	**94.1**	94.7	94.7	94.2
Promoters	105	73.6	**61.3**	**76.4**	76.4	**70.8**

Table 4. Progression from FOIL to HYDRA. The figures in the third column are the accuracy of FOIL and the figures in the remaining columns are the increase or decrease in accuracy when compared to the column immediately to the left. Figures in bold indicate statistically significant differences (using a paired t-test at 0.95) as measured with respect to the column immediately to the left. All figures represent accuracies averaged over 20 trials.

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 4). Algorithm 3 leads to a decrease in accuracy on all the variants of the KRK domain and the cancer domain. The drop in accuracy on the noise-free KRK data is due to the fact that ls-content weighs coverage more highly than fit, leading it to build fewer clauses, each of which may cover some negative examples. Interestingly, ls-content helps on the highly disjunctive DNA and the lymphography data sets. On these sets, we observed that when using information gain, there were many cases where 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 domains requiring relational concept descriptions and those requiring attribute-value concept descriptions. 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 decision components (such as 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. This has been referred to as 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

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, Dennis Kibler and Donato Malerba for reviews of this paper and to Cliff Brunk for suggestions on HYDRA

References

Ali K. and Pazzani M. (forthcoming). Reducing the small disjuncts problem by learning probabilistic concept descriptions. In T. Petsche (ed.), *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*.

Bergadano F., Matwin S., Michalski R.S. and Zhang J. Learning Two-Tiered Concept Descriptions of Flexible Concepts: The POSEIDON System. *Machine Learning, 8, 1*. 5-43.

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 Rule Induction. Submitted to the Twelfth International Joint Conference on Artificial Intelligence.

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.

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. (1991). 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.

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.