10 Reducing the Small Disjuncts Problem by Learning Probabilistic Concept Descriptions

Kamal M. Ali and Michael J. Pazzani

10.1 Introduction

Concept learners that learn concept descriptions consisting of rules 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 made by the concept description on an independent test set can be attributed to rules which were true for a small number of training examples. In noisy domains, such concept learners typically learn a few reliable rules that are true for (i.e. cover) many examples and many rules (small disjuncts) that cover few examples. This chapter concerns itself with supervised learning — that is learning in a context where the set of concepts (classes) is known *a priori* and each training example is accompanied by a class label.

This chapter presents the system HYDRA which learns a set of rules for each concept where each rule is augmented with a measure of the reliability of that rule. HYDRA is aimed at reducing the small disjuncts problem and at learning probabilistic relational concept descriptions. Probabilistic relational concepts are those that cannot concisely be described using a set of necessary and sufficient rules. For example, the concept C(X,Z) whose definition is given below, does not have a concise necessary or sufficient concept description in first-order logic.

$$p(C(X,Z) \mid E(X,Y), F(Y,Z)) = 0.8$$
$$p(C(X,Z) \mid \neg(E(X,Y), F(Y,Z))) = 0.1$$

First-order concept learners whose concept hypothesis 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 learn more and more complex approximations. HYDRA learns first-order rules and then augments them with likelihood ratios (Duda et al., 1979). HYDRA aims to estimate the reliability of a rule and then uses that to reduce the effect of small disjuncts. In section 10.1.1 we briefly explain how the relational concept learner FOIL, from which HYDRA is derived,[1] learns from data. Section 10.1.2 compares HYDRA to previous work on the small disjuncts problem and section 10.1.3 shows empirically that FOIL has a small disjuncts problem. Section 10.2 presents HYDRA and the semantics associated with our formulation of reliability. We show HYDRA learns concept descriptions that have low error rates when compared to FOIL and other algorithms. Furthermore, we present empirical evidence in section 10.3 that this increase in accuracy is due to reducing the contribution to error from small disjuncts.

10.1.1 FOIL

FOIL learns a concept description that is a conjunction of Horn clauses. Two common ways of writing such descriptions are given below.

$$(C(X,Z) \leftarrow A(X,Y) \wedge B(Y,Z)) \wedge (C(X,Z) \leftarrow D(X,Z) \wedge \neg(F(Z,Y)))$$

$$C(X,Z) \leftarrow [(A(X,Y) \wedge B(Y,Z)) \vee (D(X,Z) \wedge \neg(F(Z,Y)))]$$

We now define some terms using the above example. "$C(X,Z) \leftarrow A(X,Y) \wedge B(Y,Z)$" is a Horn clause whose head is "$C(X,Z)$" and whose body is the conjunction "$A(X,Y) \wedge B(Y,Z)$". C is called the target concept — that is, the concept for which we wish to learn a concept description. The other predicates (A,B,D and F) are called background concepts. Syntactic constructs such as $A(X,Y)$ when they appear in the body of a Horn clause are called literals. The second way of writing concept descriptions also illustrates the source of the term "small disjuncts problem". In this notation, the body of the concept description is a disjunction consisting of disjuncts that correspond to the bodies of Horn clauses.

FOIL views concepts as relations. An extensional definition of a concept C with arity n consists of all n-tuples that are in the relation C. n-tuples that are in a given relation (i.e. are members of that relation's extension) are called positive examples of that relation and n-tuples that are not in the relation are called negative examples of that relation. A tuple is a sequence of terms. We define "component" to refer to a term in a tuple. The input to FOIL consists of full extensional definitions of the background relations

[1]HYDRA is derived from FOIL via FOCL (Pazzani & Kibler, 1991). FOCL added to FOIL the ability to use a prior domain theory.

and some positive and negative examples of the target relation. The idea is to learn a concept description, using the limited numbers of positive and negative examples, that will be accurate in classifying future examples.

FOIL uses the set of background relations to build a concept description such that each positive training example of the target relation is covered[2] by one or more clauses and no negative example is covered by any clause. FOIL uses a separate and conquer approach[3] 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.

The body of each clause is learned 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 head of the clause 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 $X_1, \ldots, X_m \subseteq V_1, \ldots, V_n$.[4] We will refer to the sequence $(X_1, \ldots, X_m)$ as a variablization.

FOIL ranks each candidate literal by measuring the information (Shannon & Weaver, 1949) that would be gained if that literal was conjoined with the body of the current clause. If p denotes the number of positive examples covered by the clause before incorporation of 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, then the information gain attributed to that literal is defined by Quinlan (1990) as

$$\begin{aligned}\text{information-gain}(p, n, p^+, n^+) = p^+[&\text{information}(p^+, n^+) \\ &-\text{information}(p, n)]\end{aligned}$$

where

[2]If we replace the variables in the head of the clause by the components of the tuple to obtain a binding list and then if the instantiated body of the clause is provable, we say that the clause covers the example.

[3]The separate and conquer approach is slightly different from the divide and conquer approach used in learning decision trees (Quinlan, 1986) in that it is only concerned with modelling positive examples. This also implies that FOIL is only able to learn in situations where there are two classes - a concept and its complement.

[4]Literals where more than one of the map to the same are included in this framework.

$$\text{information}(a, b) = \log_2 \left(\frac{a}{a + b} \right)$$

After considering each candidate literal, the candidate literal providing the greatest information gain is conjoined to the body of the clause. The positive and negative examples covered by the new clause are then presented for learning the next literal in the clause. If no negative examples are covered by the new clause, that clause is considered complete. FOIL then removes the positive examples covered by that clause before proceeding to learn the next clause.

Having built a concept description, FOIL classifies an example by checking if any of the clauses are true for the example. If at least one clause is true for the example, FOIL classifies the example as a member of the target concept. If not, it uses the closed world assumption and classifies the example to the complement of the target concept. Thus, FOIL is limited to domains with two concepts which are complements of each other. 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, a learner could learn 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 generality.

HYDRA uses the same separate and conquer control strategy as used in FOIL but assumes the data are partitioned into n concepts and it learns a concept description for each concept. In addition, it attaches weights in the form of likelihood ratios to each clause. Thus, HYDRA does not assume that the examples must be partitioned into only two concepts. We discuss HYDRA in detail in section 10.2.

10.1.2 Previous Work

The work of Holte et al. (1989) demonstrates that various attribute-value learning systems on several domains are prone to the small disjuncts problem. One obvious approach to deal with this problem is to delete disjuncts that cover fewer than some preset number of examples. Holte et al. (1989) rejected this approach because it may delete disjuncts that are part of the target concept description but that cover a small fraction of the positive examples. 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 number of training examples lowers the error rates of small disjuncts. They test their method on the KPa7KR chess end-game task and find that by using a switch-over threshold of 9 examples, the overall error rate is reduced from 7.2% to 5.2% and that using a switch-over 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 10.2.

10.1.3 Evidence of the Small Disjuncts Problem

Following the definition of the small disjuncts problem as given in section 10.1.2, figure 10.1 illustrates the correlation between cumulative matches over the test set and the cumulative distribution of false positive classifications for FOIL on the "illegal" (King-Rook-King) chess problem (Muggleton et al., 1989). 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 disjuncts that match small proportions of the test examples contribute disproportionately to false positive errors. However, this problem is less severe for HYDRA because it assigns smaller weights to disjuncts that cover few examples and are likely to be unreliable. The line for HYDRA in figure 10.1 corresponds to disjuncts learned for both concepts in this domain. In order to produce figure 10.1, for each clause (disjunct), 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 $\angle\text{clause, matches, errors}\rangle$ which were sorted in ascending order by matches. Then we used the algorithm shown in table 10.1 to produce figure 10.1.

The pattern in figure 10.1 is repeated reliably on several noise-free 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 a cumulative error distribution indicated by the diagonal line in figure 10.1.

Figure 10.1
Clauses from FOIL that cover (match) only 20% of examples from the test set generate much more than 20% of the false positives (errors of comission). The problem is reduced for HYDRA. Both algorithms were trained on 160 examples from the "illegal" chess domain (Muggleton et al., 1989) with 5% tuple noise added. A level of 5% tuple noise means that for each component of a tuple, there is a 0.05 probability that the value of that component will be replaced by a random value drawn according to the uniform distribution from the set of legal values for that component. Each curve represent the average of 20 independent trials.

Table 10.1
Pseudo-code of the algorithm used to generate figure 10.1.

```
current-matches = matches of head element of $S$
for $\langle$clause, matches, errors$\rangle \in S$ do begin
    if matches $\neq$ current-matches begin
        plot(cumulative-matches/total-matches, cumulative-errors/total-errors)
    end cumulative-matches = cumulative-matches + matches
    cumulative-errors = cumulative-errors + errors
end plot(cumulative-matches/total-matches, cumulative-errors/total-errors)
```

One possible explanation for the fact that the error contribution of small disjuncts is greater than the error contribution of large disjuncts may simply be due to the relatively large number of small disjuncts learned. Our analysis compensates for this factor by comparing the proportion of false positive errors made by a set of disjuncts compared to the proportion of test examples covered by that set of disjuncts. We find that disjuncts that covered only a few *training examples* may jointly cover only 20may be jointly responsible for much more than 20positive errors. Our hypothesis then is that the small disjuncts problems can be addressed by assigning lower classification weights to small disjuncts.

10.2 Probabilistic Relational Concept Descriptions

Our motivation for developing HYDRA derives from two considerations. First, "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 descriptions end up over-fitting the data. Second, we hope to reduce the small disjuncts problem by attaching some form of classification reliability measure to each clause.

10.2.1 Knowledge Representation and Classification

Here we discuss knowledge representation used to represent concepts and how the resulting concept descriptions are used in classification. The method of learning such concepts is presented in section 10.2.2. HYDRA differs from FOIL because FOIL assumes the examples are partitioned into two classes but HYDRA assumes they may be partitioned into two or more classes (concepts). Therefore, there may be many target concepts for HYDRA, each one corresponding to a relation. (This set of concepts is disjoint with the set of background concepts.) HYDRA represents each concept by a conjunction of Horn clauses. Each clause has an associated weight representing the *degree of logical sufficiency (ls)* (Duda et al., 1979) which that clause has for the concept represented by the head of the clause. A couple of such clauses for different concepts are shown below.

$a(X, Y) \wedge b(Y, Z) \rightarrow \text{Class}_i(X, Z)\ [ls = 3.5]$

$c(X,Y) \wedge d(Y,Z) \rightarrow \text{Class}_j(X,Z)$ $[ls = 4.0]$

Let $\text{clause}_{i,j}$ denote the body of the (i,j)-th clause of the i-th concept. Given a set of positive training examples $(+)$ and negative training examples $(-)$ and a random variable t representing an example, the *ls* is defined as

$$ls = \frac{p(\text{clause}_{i,j}(t) = \text{true} \mid t \in +)}{p(\text{clause}_{i,j}(t) = \text{true} \mid t \in -)}$$

To estimate the reliability of a clause, we chose to use *ls* rather than apparent accuracy (accuracy over the training data) because Muggleton et al. (1992) have shown that coverage (the number of training examples covered by a clause) is a better indicator of the accuracy on future test examples than apparent accuracy and in our earlier work (Ali & Pazzani, 1993) we have empirically shown that *ls* is a better measure of clause reliability than coverage.

After learning one concept description per concept, a test example is classified as follows. For each concept, we want to estimate the probability that that test example belongs to the concept given that it has satisfied some clause of that concept (an example satisfies a clause if that clause covers the example). In order to do this, for each concept, considering only clauses that are satisfied by the current test example, we choose the clause with the highest *ls* value.[5] The clause with the highest *ls* value is chosen because it is considered the most reliable indicator of the concept, based on the training data. We do this for each concept and assign the test example to the concept whose representative clause has the highest ls value. 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 concept is to multiply together the *ls* values of all the clauses within each concept, 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 methods are empirically compared in table 10.2.

[5] 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.

10.2.2 Learning in HYDRA

HYDRA differs from FOIL in three major ways. First, HYDRA learns a concept description for each concept. Second, HYDRA associates an estimate of the degree of logical sufficiency with each learned clause. Third, HYDRA uses a candidate literal ranking metric (*ls*-content) that is aimed at learning probabilistic concept descriptions rather than the information gain metric which is aimed at learning necessary and sufficient concept descriptions. Learning one concept description per concept 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 *ls* during learning as well as for estimating the reliability of a clause. Consider the learning process midway through learning the j-th clause for the i-th concept. Let $p_{i,j,0}$ denote the number of positive examples not covered by previous clauses. Thus, these examples are available for learning the current clause. However, because some literals may already have been added to the body of the current clause, not all of the examples may satisfy the current clause. Let $p_{i,j}$ denote the number of positive examples that satisfy the current clause. Define $n_{i,j,0}$ and $n_{i,j}$ analogously. These quantities can then be used to estimate *ls* from data using the Laplace approximation (Kruskal & Tanur, 1978):

$$p(\text{clause}_{i,j}(t) = \text{true} \mid t \in +) \approx \frac{p_{i,j} + 1}{p_{i,j,0} + 2}$$

(The justification for this approximation follows later in this section.) An analogous approximation can be made for the denominator of *ls* to yield a method of computing *ls* from data

$$ls(p_{i,j}, n_{i,j}, p_{i,j,0}, h_{i,j,0}) \approx \frac{(p_{i,j} + 1)(n_{i,j,0} + 2)}{(n_{i,j} + 1)(p_{i,j,0} + 2)}$$

Using this definition, *ls-content* is defined as

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

where α is a parameter to the system.

Using this metric, HYDRA compares the value of *ls-content* before addition of a literal to the body of the clause to the value after addition of the literal. If there are no literals that cause an increase in *ls-content*, HYDRA completes the clause, otherwise it resets $p_{i,j}$ and $n_{i,j}$ to reflect the numbers of examples covered by the clause with the new literal conjoined. The parameter *ls-content* trades off discriminability against coverage as did information content, but it is weighed more in favor of coverage as is necessary in probabilistic domains and when learning from noisy data. Setting α to 0 causes HYDRA to build many clauses, none of which cover many examples. Setting $\alpha = 1$ causes HYDRA to build no clauses, effectively reducing HYDRA to guessing the concept with the highest prior probability estimate. For the experiments in section 10.3, we set α to a neutral intermediate value of 0.5.

After all the clauses have been learned, HYDRA forms an estimate of the degree of logical sufficiency, $ls_{i,j}$, associated with each clause using the entire set of positive training examples (+) and the negative training examples ($-$),

$$ls_{i,j} = \frac{p(\text{clause}_{i,j}(t) = \text{true} \mid t \in +)}{p(\text{clause}_{i,j}(t) = \text{true} \mid t \in -)}.$$

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

$$p(\text{clause}_{i,j}(t) = \text{true} \mid t \in +) \approx \frac{p_{i,j} + 1}{p_i + 2}$$

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

$$ls_{i,j} = ls(p_{i,j}, n_{i,j}, p_i, n_i) \approx \frac{(p_{i,j} + 1)(n_i + 2)}{(n_{i,j} + 1)(p_i + 2)}$$

This is the weight that HYDRA attaches to a clause. It is different from the likelihood ratio used during learning in that the conditional probabilities are taken with respect to the entire training set, not just the subset uncovered by the previous j clauses.

Note that the Laplace ratio also has the convenient property that it does not assign a *ls* of infinity to a clause that may cover just 1 positive and 0 negative training examples. This is convenient because a *ls* of infinity means satisfaction of that clause is totally sufficient to classify the test example as a member of the concept associated with that clause.

10.3 Experimental Results

In this section, we show that the three 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. 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 (false negatives) 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 concept in the training data. If a test example satisfies clauses from more than one concept, the test example is classified to the concept 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.

10.3.1 Description of the Domains

We ran experiments on six variants (see table 10.2) 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 labeled *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 $illegal(V_1, \ldots, V_6)$, 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.

10.3.2 Experimental Comparison of HYDRA and FOIL

Table 10.2 shows that a method of assigning lower weights to less reliable clauses in noisy domains and even in the noise-free 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, 20randomly 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 on lymphography. It also does better than Reduced-Error Pruning applied to FOIL (Brunk & 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 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.

Table 10.2
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 occurring concept. 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.

Task	Number of training examples	FOIL accuracy	Optimistic likelihood Estimation	Pessimistic likelihood Estimation
Illegal with 20%	160[a]	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**[b] (4.2)	**79.8** (5.4)	**78.6** (5.8)
KPa7KR	200	90.3 (2.5)	**94.7** (1.1)	
Breast Cancer	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)

[a]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.
[b]This accuracy is for MC-FOIL because FOIL cannot be run on domains with more than two classes and the lymphography data set contains four classes.

Figures 10.1 and 10.2 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 that together cover 20% or less of the test examples made 80% of the false positive classifications. Only one "match" is attributed per test example. For FOIL, this attribution is made to the first clause (in order of learning) 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 cover a small percentage

Figure 10.2
Comparison of the cumulative error distribution of FOIL and HYDRA to the ideal cumulative error distribution on the DNA promoters domain. A comissive error is a false positive classification.

of the examples from the test set are responsible for a disproportionate percentage of the total false positive classifications. 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 10.2) as well as the distribution of error.

10.3.3 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 α parameter. figure 10.3 plots how varying a affects accuracy. figure 10.3 presents preliminary evidence that the best value of α is one that neither over-fits the data ($\alpha = 0$) nor one that underfits the data ($\alpha = 1$).

Figure 10.3
Variation in accuracy as a function of varying the degree to which 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.

10.3.4 Partial Clauses

Concept descriptions learned by HYDRA and FOIL suffer from large numbers of false negatives (errors of omission) when learning M of N concepts (Spackman, 1988) or when learning highly disjunctive concepts. On the DNA promoters domain for example, all clauses of all concept descriptions learned by HYDRA failed to cover 13% of the test examples from the promoters concept and 21% of the test examples of the non-promoters concept. In such cases HYDRA is forced to guess the most frequent concept. A better alternative is to determine if the clause nearly covers an example. We implemented this idea by adding clauses that are derived from the clauses already learned by HYDRA. For example, the clause $a(X,Z) \wedge b(Y,X) \wedge c(X,X) \rightarrow \text{concept}(X,Y)$ would give rise to the fol-

Table 10.3
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.

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)

lowing additional clauses:

$$a(X,Z) \wedge b(Y,Z) \rightarrow \text{concept}(X,Y)$$
$$a(X,Z) \rightarrow \text{concept}(Y,Z)$$

These clauses serve as backup in case none of the clauses covers a test example. These clauses, which we will refer to as "partial clauses", tend to cover larger regions of the instance space but tend to do a poorer job of discriminating positive from negative examples (and hence, have lower ls values). Table 10.3 gives a comparison of accuracies with and without partial clauses.[6] Table 10.3 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

[6]Average accuracies shown in table 10.3 are slightly different from those shown in table 10.2 because the tables used different sets of runs of HYDRA.

test examples that failed to match any clause of any concept description. As expected, addition of partial clauses reduces components of these vectors.

10.4 Conclusions and Future Work

We have presented a method using maximum likelihood estimation for reducing the small disjuncts problem and thereby increasing classification 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 concept descriptions per concept and then combine evidence from these models. This approach 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.

You have two citations, Buntine and Holte that seem to conflict same conference name, two different years and cities. What are the correct citations?
Buntine W. 1991. Classifiers: A Theoretical and Empirical Study. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Sydney, Australia: Morgan Kaufmann.

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.

IV Theory