On Learning Multiple Descriptions of a Concept

Kamal Ali, Clifford Brunk and Michael Pazzani
Department of Information and Computer Science,
University of California, Irvine, CA, 92717
{ali,brunk,pazzani}@ics.uci.edu
714-725-3491, 714-856-5888

July 15, 1994

Abstract

In sparse data environments, greater classification accuracy can be achieved by learning several concept descriptions of the data and combining their classifications. Stochastic search is a general tool which can be used to generate many good concept descriptions (rule sets) for each class in the data. Bayesian probability theory offers an optimal strategy for combining classifications of the individual concept descriptions, and here we use an approximation of that theory. This strategy is most useful when additional data is difficult to obtain and every increase in classification accuracy is important. The primary result of this paper is that multiple concept descriptions are particularly helpful in "flat" hypothesis spaces in which there are many equally good ways to grow a rule, each having similar gain. Another result is experimental evidence that learning multiple *rule sets* yields more accurate classifications than learning multiple rules for some domains. To demonstrate these behaviors, we learn multiple concept descriptions by adapting HYDRA, a noise-tolerant relational learning algorithm.

Academic Track. Area: Machine Learning

1 Introduction

We present a method for learning multiple concept descriptions (rule sets) for each class[1] in the data and for combining classifications produced by these concept descriptions. Learning multiple descriptions of the same class can be used as a tool to increase accuracy in domains where additional data are hard to obtain and where small improvements in accuracy can lead to large savings in cost. Although many systems learn attribute-value descriptions, there exist many concepts which are not easily represented in attribute-value form but which can be succinctly represented in relational form. In order to learn these concepts, we use the relational learning algorithm HYDRA (Ali & Pazzani, 1993; Ali & Pazzani, 1992). We present results on the following relational problems: predicting finite-element mesh structure (Dolsak and Muggleton, 1991), learning the King-Rook-King concept (Muggleton *et al.*, 1989), classifying a document block (Esposito *et al.*, 1993) and predicting whether a person is required to make payments on their student loan (Pazzani and Brunk, 1991). The document block classification domain requires learning a concept description containing recursion. Although previous work has shown that learning multiple concept descriptions increases accuracy for other kinds of concept descriptions (e.g. decision trees, Buntine 1990) there has been no prior work in learning multiple concept descriptions for relational problems or in learning multiple concept descriptions, each one being a rule set.

The learning task addressed in this paper takes as input a collection of examples belonging to a set of specified classes which partition the example space and it also takes as input a set of *background relations* for which full definitions are provided to the learning algorithm. The task then is to build a concept description for each class using combinations of the background relations. An example is a sequence

[1] We will use the terms "class" and "concept" interchangeably.

of ground terms (such as (1,foo)); examples of all classes are of the same length (arity). After learning, the task is to take a novel test example and find which class it belongs to. This framework is a generalization of the work of Quinlan (1990) on learning relational descriptions. In this paper we will refer to first-order Horn clauses (e.g. Lloyd, 1984) as rules. A first-order Horn clause such as *class-a(X, Y)* $\leftarrow b(X), c(Y)$ consists of a head (*class-a(X, Y)*) and a body which is a conjunction of *literals* $(b(X), c(Y))$[2]. The arrow signifies logical implication. In this paper, we will refer to a list of such rules, each with the same head, as a *concept description* or *rule set*. A collection of such concept descriptions, one for each class in the data, is called a *model*. Classification of a novel test example (in the case of a single model) proceeds as follows: an attempt is made to resolve the test example with each rule for every class. Hopefully, only rules for a single class will successfully resolve with the example and so it will be classified to that class. Later in the paper we discuss what to do if it resolves with rules of more than one class or if it does not resolve with any rules of any class.

In order to learn multiple concept descriptions for each class, we have modified HYDRA which we have previously shown can learn accurately in noisy, relational "real world" domains. Most learning programs, including HYDRA, learn just one model of the data. In this paper we present HYDRA-MM which learns multiple models of the data, each model being a collection of concept descriptions (Figure **??**). Therefore, HYDRA-MM also learns multiple concept descriptions for a given class. Learning multiple concept descriptions and combining their classifications is one way to improve accuracy on "real world" concepts which may not be representable by a set of rigid rules (Smith and Medin, 1981) but, rather, need evidence combination from many sources. Multiple descriptions of a concept offer

[2] b and c are background relations whose definition is presented to the learner.

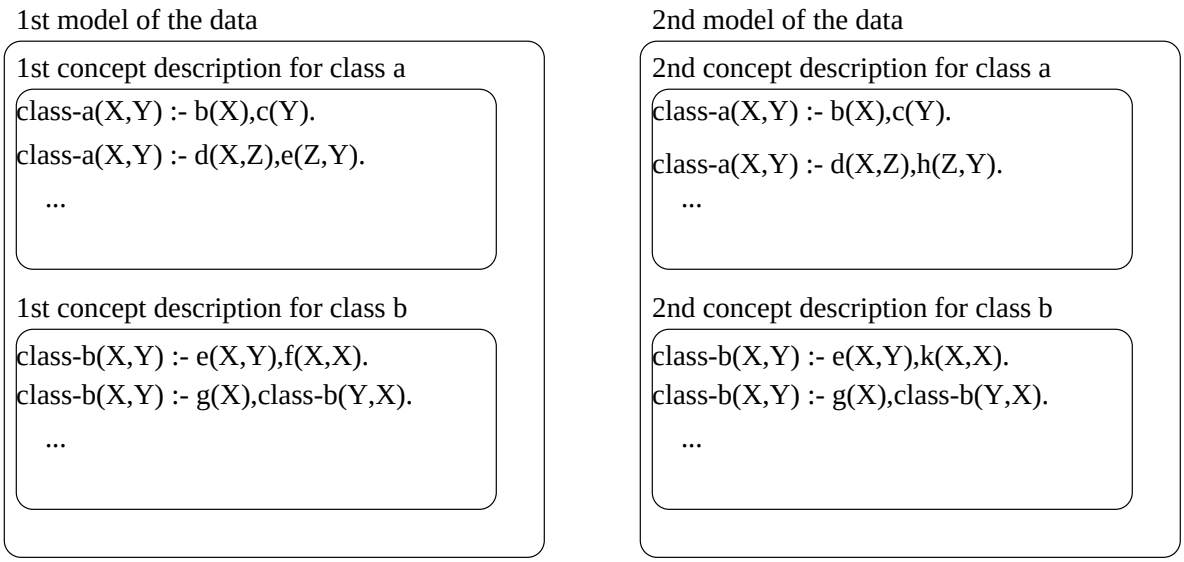


Figure 1. Multiple rule sets: most algorithms learn one model for the data. HYDRA-MM learns several (each consisting of a rule set) and combines classifications between rule sets for a given class.

one way of making decisions based on evidence combination. Figure **??** for instance, shows 2 models learned from data consisting of 2 classes. Note that the rules are first-order Horn clauses and can be recursive. Figure **??** shows that typically, concept descriptions for a given class are correlated, and may involve replacing a literal such as $e(Z,Y)$ with some other literal which approximates the first: $h(Z,Y)$. Of course, this does not preclude the possibility of greater variation between the concept descriptions for a given class.

The primary goal of this research is to characterize the conditions under which learning multiple concept descriptions from training examples is beneficial to classification accuracy on novel, test examples. Another objective is to create a useful tool by applying Bayesian probability theory which postulates that for optimal accuracy, one should make classifications based on voting among all possible concept descriptions in the hypothesis space, each one weighted by its posterior probability.

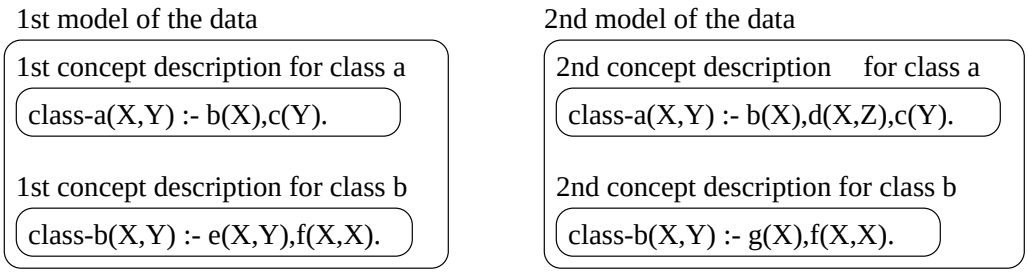
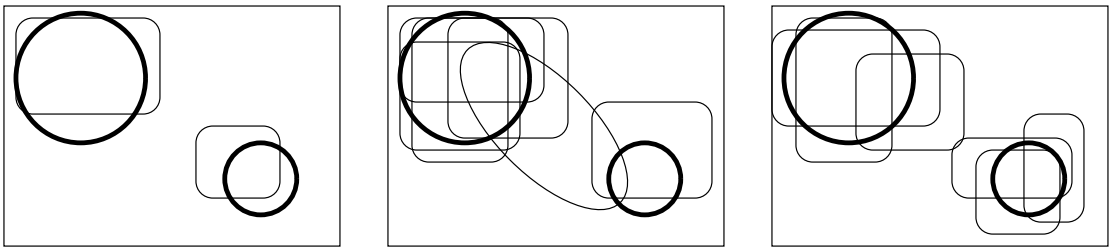


Figure 2. Multiple rules approach learns multiple descriptions for each concept. Because those descriptions consisted of single rules rather than rule sets, they cannot easily be applied to concepts containing multiple disjuncts.

In practice, however, it is only possible to learn a few "good" (high posterior probability) descriptions and then make classifications based on voting among those descriptions. Therefore, we also investigate alternative methods for generating multiple concept descriptions. We also experimentally show that learning multiple rule sets yields higher accuracy than learning multiple rules on a domain in which the classes consist of multiple disjuncts. For this paper, a class is said to contain multiple disjuncts, if (with respect to the hypothesis language being used) the target description that the algorithm is trying to learn can only be modeled with more than one rule.

2 Issues in multiple concept descriptions

In this paper we argue that for concepts containing many disjuncts it is more appropriate to learn multiple concept descriptions where each consists of a rule set (*multiple rule sets*) rather than a rule (*multiple rules*). Two objectives motivate our work on learning multiple rule sets. First, to ensure that each disjunct in the target concept is modeled by more than one rule. Second, two ensure that some rules are learned for minor disjuncts. Figure 3 illustrates a concept containing a major disjunct (large dark circle) and a minor disjunct (small dark circle). Light



Single Model Multiple Rules Multiple rule sets

Figure 3. Comparison of 3 algorithms trying to learn on a domain where the first class consists of 2 disjuncts (dark circles). The area outside the dark circles corresponds to the other class in this domain. Light lines show the coverage of rules learned by the 3 algorithms.

lines indicate learned rules that try to approximate the underlying disjuncts. The leftmost figure illustrates what is learned using a separate and conquer technique (e.g. FOIL, Quinlan, 1990) which learns an approximation (rule) to the first disjunct and then removes training examples covered by that rule in order to learn subsequent rules. The middle figure illustrates what may be learned by a multiple rules approach told to learn 6 rules. Note that the multiple rules approach (e.g. Kononenko and Kovacic, 1992) does not remove training examples covered by previously learned rules in order to learn other rules. It uses a metric which gives a greater reward for covering previously uncovered examples. Even though the approach of Quinlan and the approach of Kononenko and Kovacic both produce a set of rules, they are used and interpreted in different ways. In Quinlan's method, each rule attempts to model one disjunct in the target concept. In what we are calling the multiple rules approach (e.g. Kononenko and Kovacic), each rule models the entire target concept. This kind of multiple rules approach does not meet the second objective - attempts to model the minor disjunct are distracted by training examples from the major disjunct. Finally, the rightmost figure illustrates what is ideally learned by HYDRA-MM - which applies a separate and conquer approach

many times. Figure 3 shows HYDRA-MM has learned 3 concept descriptions for the first class (areas inside dark circles), each concept description being learned by a separate and conquer approach. The figure shows that each concept description consists of 2 rules. Note that this meets both of the objectives outlined above: each disjunct is modeled multiple times and attempts to model the small disjunct are less influenced the major disjunct.

Previous methods for generating multiple concept descriptions include beam search (Smyth *et al.*, 1992), stochastic search (Kononenko *et al.*, 1992; Buntine, 1990), user-intervention (Kwok and Carter, 1990) and n-fold cross-validation (Gams, 1989). In the approach of Kwok and Carter, the user examines the best candidate splits for growing a decision tree and can then choose to use some of them to grow separate trees. Gams partitions the training set into 10 equal subsets and then uses leave one out on the subsets to generate 10 rule sets. In a final pass he uses all the data to learn, only retaining rules that appear in a significant proportion of the 10 rule sets. Except for the stochastic search methods, these methods do not generate concept descriptions that are independent of each other (conditional on the training data) so they cannot use several theoretically sound evidence combination methods. Kononenko and Kovacic use the naive Bayesian combination method even though their rules do not meet the assumptions under which that method can be used.

Evidence combination methods previously used include naive Bayesian combination (Kononenko, Smyth) and combination according to the posterior probability of the concept description (Buntine). In this paper we will compare these two methods. We will use the odds[3] form of naive Bayesian combination because HYDRA associates likelihood odds ratios (Duda *et al.*, 1979) with rules rather

[3]The odds of a proposition with probability p are $p/(1-p)$.

than class probability vectors.

3 Theoretical background

The Bayesian approach postulates that assuming we want to maximize some utility function u that depends on some action and an incompletely known state of the world, we should take the action A that maximizes the Bayesian expected utility where the expectation is summed over all possible worlds H. In our case, this implies that we should make the classification (A) whose expected accuracy (u) is maximal, with the expectation being taken over all possible hypotheses in the hypothesis space of the learning algorithm. In practice, we only compute this expectation over a small set of "good" hypotheses. Using the notation in Buntine (1990), let c be a class, $\mathcal{T}$ be the set of hypotheses the learning algorithm has produced, x be a test example and $\vec{x}$ denote the training examples. Then, we should assign x to c that maximizes

$$pr(c|x,\mathcal{T}) = \sum_{T in \mathcal{T}} pr(c|x,T)pr(T|\vec{x}) \tag{1}$$

A rule set can be viewed as a decision tree with compound tests at the nodes, each test corresponding to the body of one clause. This allows us to use equation **??** to compute the probability of a rule set. For a rule, $pr(c|x,T)$ can be roughly approximated from the training accuracy of that rule. $pr(T|\vec{x})$ for a binary tree structure is defined as

$$pr_2(n_{1,1},n_{2,1},n_{1,2},n_{2,2}) \stackrel{\text{def}}{=} pr(T|\vec{x}) \propto pr(T) \times \prod_{k=1}^{V} \frac{B(n_{1,k}+\alpha_1, n_{2,k}+\alpha_2)}{B(\alpha_1,\alpha_2)} \tag{2}$$

where $pr(T)$ is the prior of the tree, B is the beta function[4], V is the number of leaves of the tree, α_1 and α_2 are parameters and $n_{i,j}$ denotes the number of training examples of class i at the j-th leaf of the tree. pr_2 is the probability of a

[4]For integer, strictly positive p and n, $B(p,n) = (p-1)!(n-1)!/(p+n-1)!$

binary subtree with 2 leaves and uniform prior (it is used in section **??** to prove a property of the Bayes gain metric).

Letting $n_{1,ij}$ and $n_{2,ij}$ respectively denote the number of positive and negative examples covered by the j-th rule of the i-th class, and V the number of rules in a rule set for class i, we can use equation **??** to compute the posterior probability of a rule set learned by HYDRA.

Buntine observes that being able to compute the posterior probability of a tree means one should use the posterior probability as a gain metric to search for the most probable trees (although both our systems are limited by a one step lookahead). To compute the gain of adding a test to a tree (or a literal to a clause) then, one computes the difference between the probability after addition and the probability before addition (using equation **??**).

4 Separate and Conquer Strategy

HYDRA uses a separate and conquer control strategy based on FOIL (Quinlan, 1990) in which a rule is learned and then the training examples covered by that rule are separated from the uncovered examples for which further rules are learned. A rule for a given class such as *class-a(V1,V2)* is learned by a greedy search strategy. It starts with an empty clause body, which in logic is equivalent to the rule *class-a(V1,V2)* $\leftarrow$ *true* and so covers all remaining positive and negative examples. Next, the strategy considers all literals that it can add to the clause body and ranks each by the information gained (Quinlan, 1990) by adding that literal. Briefly, the information gain measure favors the literal whose addition to the clause body would cover many positive examples and exclude many negative examples. For example, if the algorithm is considering adding a literal to the rule *class-a(V1,V2)* $\leftarrow$ *d(V1,V3)* and the set of background relations is {*b*,*d*}

Table 1. Pseudo-code for FOIL.

```
FOIL(POS,NEG,Metric): (Metric is usually Information Gain)
  Let POS be the positive examples.
  Let NEG be the negative examples.
  Set NewClauseBody to empty.
  Until POS is empty do:
      Separate: (begin a new clause)
      Remove from POS all positive examples that satisfy the NewClauseBody.
      Remove from NEG all negative examples that satisfy the NewClauseBody.
      Reset NewClauseBody to empty.
      Until NEG is empty do:
          Conquer: (build a clause body)
              Choose a literal L using Metric
              Conjoin L to NewClauseBody.
              Remove from NEG examples that do not satisfy NewClauseBody.
```

where *b* is of arity 1 and *d* is of arity 2, the literals it will consider are {*b(V1), b(V2), b(V3), d(V1,V1), d(V1,V2), d(V1,V3), d(V1,V4), d(V2,V1), d(V2,V2), d(V2,V3), d(V2,V4), d(V3,V1), d(V3,V2), d(V3,V3), d(V3,V4), d(V4,V1), d(V4,V2), d(V4,V3)*}. The strategy keeps adding literals until one of the two following conditions occurs: the clause covers no negative examples or there is no candidate literal with positive information gain. Examples covered by the clause are removed from the training set and the process continues to learn on the remaining examples, terminating when no more positive examples are left.

5 HYDRA

HYDRA is derived from the inductive learning component of FOCL (Pazzani and Kibler, 1991) which added to FOIL the ability to learn using intensionally defined prior background knowledge. HYDRA differs from FOCL in three important ways.

1. HYDRA learns a set of rules for each class so that each set can compete to classify test examples.
2. HYDRA attaches a measure of classification reliability to each clause.
3. The metric used by HYDRA to search for literals trades off positive training coverage against training accuracy more in favor of coverage than is done by the information gain metric. Trading off in this manner discourages overfitting the data.

Knowledge Representation- Clauses learned by HYDRA are accompanied by a likelihood odds ratio (our choice for measuring clause reliability). The likelihood odds ratio of the *ij-th* clause of the *i-th* rule set is defined as follows:

$$ls_{ij} = \frac{pr(clause_{ij}(\tau) = true \mid \tau \in Class_i)}{pr(clause_{ij}(\tau) = true \mid \tau \notin Class_i)} \quad (3)$$

where τ represents an arbitrary example. The likelihood odds ratio is a generalization of the notion of sufficiency that the body of a clause has for the head of that clause. Muggleton *et al.* (1992) show that training coverage is a more reliable indicator of the true accuracy of a clause than apparent accuracy measured from the training data. We choose to use LS as a reliability measure because it has the flavor of measuring both coverage and accuracy. Between 2 clauses covering n_1 positive and 0 negative, and n_2 positive and 0 negative respectively, the LS of the first clause will be higher if $n_1 > n_2$. Furthermore, if the clauses cover some negative examples, LS is equivalent to measuring accuracy, normalized with the prior probabilities of the classes.

HYDRA classifies an example by sorting all clauses (of all rule sets) according to their LS values and then assigning the test example to the class associated with the first satisfied clause. When a test example satisfies no clause of any class, HYDRA guesses the most frequent class.

Table 2. Pseudo-code for HYDRA

```
HYDRA(Metric,POS_1,...,POS_n):
    For i in classes 1 to n do
        POS = POS_i
        NEG = (POS_1 union ... POS_n) - POS_i
        ConceptDescription_i = FOIL(POS,NEG,Metric)
        For rule R in ConceptDescription_i do
            Augment R with its LS
```

Learning in HYDRA- HYDRA uses the same separate and conquer strategy used in FOCL. However, HYDRA uses the *ls-content* metric which is defined as follows. If the *j-th* clause covers p positive examples[5] and n negative examples and there were p_j and n_j examples uncovered after the first $j-1$ clauses, its *ls-content* is defined as:

$$\textit{ls-content}(p, n, p_j, n_j) = ls(p, n, p_j, n_j) \times p$$

Using this metric HYDRA compares the *ls-content* before addition of a literal to a clause to that after addition of the literal. When there are no literals that cause an increase in *ls-content* or if the clause no longer covers any negative examples, HYDRA moves onto the next clause. HYDRA learns more accurate rule sets using *ls-content* than it does using information gain (Ali and Pazzani, 1993).

After all the clauses have been learned, HYDRA forms an estimate of the likelihood ratio ls_{ij} associated with clause ij of concept i (this time, it uses the entire training data, not just the examples uncovered by previous clauses). We use Laplace's law of succession to compute estimates for the two probabilities in ls_{ij} giving,

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

[5]Because clauses can introduce variables not found in the head of the clause, each example may become "extended" through new variables. In HYDRA, all counts of examples are done in the unextended space.

where p_0 and n_0 respectively denote the numbers of positive and negative training examples (of $Class_i$) and p and n denote the numbers of examples covered by the clause. Clauses with $ls_{ij} \leq 1$ are discarded.

5.1 HYDRA with multiple concept descriptions

HYDRA generates one rule set per class whereas HYDRA with multiple concept descriptions (HYDRA-MM) generates many rule sets for each class. Stochastic search is used to generate multiple concept descriptions. During stochastic search in HYDRA-MM instead of choosing the literal which maximizes the literal selection metric, it stores the *MAX-BEST* top literals (ranked by gain) and then chooses among them with probability proportional to their gain. Recall that Bayesian theory recommends making classifications based on voting from all concept descriptions in the hypothesis space but in practice we want to find a few "good" concept descriptions. Stochastic search is one way of trying to find these "good" descriptions. Ideally, one would want the n most probable concept descriptions with their probability being evaluated globally, but the search would be intractable. Instead, we settle for a tractable (greedy search) approximation to the theoretically optimum goal. For the experiments described in section **??**, we set *MAX-BEST* to 2. Note that our method for probabilistically generating concept descriptions differs from Kononenko *et al.* (1992) in that we do not initialize each clause with a randomly selected literal. It also differs in that we are greedily searching for the most probable descriptions whereas their work involves maximizing an ad-hoc metric.

Classification with multiple concept descriptions learned using *ls-content* proceeds as follows for some class $Class_i$. For rule set M_{ij} of $Class_i$, let ls_{ij} denote the LS of the clause with the highest LS value (taken only over clauses satisfied

by the current test example). We combine these LS values using the odds form of Bayes rule (Duda *et al.*, 1979) in which the prior odds of a class are multiplied by the LS's of all satisfied clauses. If M_i denotes the collection of rule sets for the i-th class and M_{ij} denotes the ij-th rule set, then

$$O(Class_i|M_i) \propto O(Class_i) \times \prod_j O(Class_i|M_{ij}) \quad (5)$$

$O(Class_i)$ are the prior odds of $Class_i$ and we use ls_{ij} for $O(Class_i|M_{ij})$ because for a given test example, each rule set is represented by the most reliable clause - the satisfied clause having the greatest LS. The test example is assigned to the class maximizing $O(Class_i|M_i)$.

We will compare HYDRA-MM using *ls-content* with HYDRA-MM using the Bayes gain function - which we now define. As Bayesian theory prescribes voting among concept descriptions with weight proportional to the posterior probability of the concept description, it makes sense to search for concept descriptions with high posterior probability (Buntine, 1990). This implies that the function pr_2 (equation **??**) should be used as a literal selection metric for learning clauses. The problem with using pr_2 for learning clauses in HYDRA-MM is that pr_2 is symmetric so that a clause covering 5 (p) out of 10 (p_0) positive and 1 (n) out of 10 (n_0) negative examples will receive the same score as one covering 5 out of 10 negative and 1 out of 10 positive examples. Thus, we use a modified pr_2 function: one in which pr_2 is assigned 0 if $p/n \leq p_0/n_0$. We chose a value of 1 for α_1 and α_2 because that value is consistent with the weight to the prior used in Laplace's law of succession.

For our work with multiple concept descriptions involving the Bayesian gain metric, we followed the approach for *ls-content* and decided that for a given test example, each rule set would be represented by only one clause. Accordingly, we choose the satisfied clause whose product of accuracy and probability is the

largest. Such a choice is made for each rule set for a given class and then all these products (within the class) are combined according to equation ?? to yield the posterior probability for that class. HYDRA-MM assigns the test example to the class maximizing equation ??.

6 Experimental Results

The main goal of our experiments was to see whether multiple concept descriptions are most useful in "flat" hypothesis spaces. Other goals were to see how increasing the number of models affects accuracy and to compare HYDRA-MM using *ls-content* to HYDRA-MM using the Bayes gain metric. Table 1 compares accuracies obtained by 1 deterministically generated concept description to those generated by stochastic hill-climbing. The concept descriptions were generated incrementally so, for instance, the 2 concept description accuracies include the model that generated the 1 concept description accuracies. We tested four variants in table 1. *Lsc* denotes learning with *ls-content* using uniform priors. *Lsc+p* denotes use of "priors" using global frequencies observed in the training data. Any differences in accuracy between these two variants was due to the influence of priors. *Bayes* denotes learning with the Bayes gain metric and voting using the accuracy of the clause and the probability of the concept description (equation ??). *Bayes-p* is a simplification in which concept descriptions vote using just the accuracy of a clause. These 2 variants used the same learned description.

We tested HYDRA-MM with the *ls-content* and the Bayesian gain metrics on the following well-known relational domains: finite element mesh structure with 5 objects (Dzeroski and Bratko, 1992), King-Rook-King illegality (Muggleton *et al.*, 1989), document block classification (Esposito *et al.*, 1993) and student loan prediction (Brunk and Pazzani, 1991). In addition, we also tested it on

the following domains traditionally regarded as "attribute-value domains". In the breast cancer recurrence prediction domain and the lymphography domain we added the relation *equal-attr(X,Y)* which allows attribute-attribute comparisons in addition to attribute-value comparisons. In the DNA promoters domain we added the predicate *y(X)* which is true if the base-pair X is a T or a C, and the predicate *r(X)* which is true if X is an A or a G. We also use the King-Rook King-Pawn (KRKP) domain which has been demonstrated to have small disjuncts (Holte *et al.*, 1989).

Representation- For the attribute-value domains we created tuples whose length matched the number of attributes. This minimized the need to introduce attributes through literals involving existentially quantified variables - a relatively expensive process. For the relational domains, we used the same representation used by previous authors.

Testing Methodology - For the finite-element mesh data set, the examples are derived from 5 objects. Dzeroski and Bratko use a "leave one object out" testing strategy in which on the i-th trial, positive examples from object i are used for testing and positive and negative examples from other objects are used for training. The fact that testing never occurs on negative examples is important in explaining some of the results on this domain. Leave-one-out testing was used on the DNA promoters domain and 10-fold cross validation was used on the document block classification domain. For the KRK data sets, we used 20 independent training and test sets. For the cancer and lymphography data sets, we used 20 partitions of the entire data, split 70% and 30%. For the students domain, we used 100 training and 900 testing examples and for the KRKP domain, we used 200 training and 1000 testing examples. In the KRK domain entries in Table 1, the first figure indicates number of training examples, the second represents the percentage of class noise

that was artificially introduced.

6.1 Discussion

The major goals of this research are to determine the conditions under which learning multiple descriptions yields the greatest increase in accuracy and to learn multiple descriptions in a way consistent with Bayesian probability theory. Table 1 shows that except for the KRK 160,20 task and some *Mesh* variants, multiple concept descriptions always increase accuracy either by a statistically significant margin or by a substantial margin (when the t-test is not applicable). Our hypothesis is that multiple concept descriptions help in domains where no one literal has much higher gain than others. To measure this "flatness" during the learning process of adding a literal to a clause, we keep track of the number of occasions on which multiple candidates tied for the best gain. Table 1 indicates that multiple concept descriptions are most helpful in the DNA promoters domain which also has a very "flat" hypothesis space with respect to the *ls-content* metric. In fact, for this domain, the fraction of occasions on which 10 or more candidates tied for the best gain was 12.2%, much more than for other domains.

In table 1, *Bayes* gets significantly higher accuracy (1 deterministic description) than *Lsc* on 4 domains, worse on 3 and the same on 2 domains. Later in this section we show that the Bayes gain metric has an undesirable property which may explain its relatively poor performance on those 3 domains. At 11 descriptions, *Bayes* does better than *Lsc* on only the KRK 160,20 task, and worse on DNA and Mesh domains. That is, as the number of descriptions is increased, there is an equalization between the algorithms. However, we observe that if an algorithm has a superior accuracy to another for 1 description, then usually it is equal or better at 11 descriptions.

Domain	Algo-rithm	Deter-ministic	1 CD	2 CDs	5 CDs	11 CDs	SIG.	ties	# Train eg.s
Promoters	Lsc	77.4	75.5	83.0	89.6	92.5	NA	24.0%	53,53
	Lsc+p	77.4	74.5	83.0	89.6	92.5	NA		
	Bayes	65.1	75.0	77.2	85.9	84.8	NA		
	Bayes-p	65.1	73.6	82.1	84.0	85.8	NA		
Lymph.	Lsc	79.6	75.5	78.9	80.1	82.3	80	21.1%	3,41,
	Lsc+p	78.3	76.4	79.5	81.1	82.4	95		54,1
	Bayes	79.3	80.5	80.5	82.0	82.8	98		
	Bayes-p	79.3	77.7	78.4	79.2	79.8	NS		
Cancer	Lsc	67.2	68.1	68.6	69.4	70.4	95	9.3%	54,127
	Lsc+p	70.4	70.7	71.7	70.9	71.1	NS		
	Bayes	71.2	71.5	71.1	71.1	71.1	NS		
	Bayes-p	71.2	70.9	71.1	72.4	72.1	NS		
KRKP	Lsc	94.5	94.1	95.0	95.7	95.7	95	18.7%	104,96
	Lsc+p	94.3	94.0	94.9	95.7	95.7	95		
	Bayes	94.6	94.3	94.8	94.8	94.9	NS		
	Bayes-p	94.6	94.0	94.8	95.6	95.7	95		
Mesh	Lsc	50.7	60.4	53.6	62.9	61.2	NA	3.9%	222,2272
	Lsc+p	32.0	22.3	15.4	38.5	52.2	NA		
	Bayes	33.5	36.7	50.7	49.3	21.2	NA		
	Bayes-p	33.5	43.0	47.0	45.9	39.1	NA		
Document	Lsc	99.2	99.2	99.6	99.6	100	NA	25.2%	22,198
	Lsc+p	98.4	98.3	99.6	99.6	100	NA		
	Bayes	98.3	97.4	98.4	99.0	99.5	NA		
	Bayes-p	98.3	97.4	98.4	99.0	99.0	NA		
Students	Lsc	82.9	83.3	85.9	88.7	89.2	99	7.3%	64,36
	Lsc+p	87.9	87.0	87.4	89.0	89.2	80		
	Bayes	86.1	85.4	86.9	88.9	90.4	99		
	Bayes-p	86.1	85.0	86.4	87.7	89.4	99		
KRK 160,20	Lsc	90.6	88.6	89.6	90.0	89.9	NS	7.2%	48,112
	Lsc+p	91.3	89.5	90.2	91.0	90.3	NS		
	Bayes	92.3	91.6	91.8	91.9	92.0	NS		
	Bayes-p	92.3	90.3	90.7	90.9	91.6	NS		
KRK 320,20	Lsc	93.3	92.5	93.6	94.4	94.7	90	5.8%	96,224
	Lsc+p	93.7	92.5	94.2	94.7	95.0	80		
	Bayes	95.5	94.9	95.3	95.6	95.5	NS		
	Bayes-p	95.5	93.9	94.2	94.8	95.2	NS		

Table 1. Accuracies obtained by 4 HYDRA variants. The numbers in the 2nd-last column give the significance level (SIG) at which 11 concept descriptions (CDs) accuracy differs from that of 1 deterministic description according to the paired 2-tailed t-test. NS - not significant, NA - t-test not applicable.

Now we examine if priors help when learning with *ls-content*. For 1 concept description, priors help significantly for the breast-cancer domain and hurt for the Mesh and students domains. At 11 concept descriptions, there is an equalization so that priors become less of an issue. We illustrate the effect of priors by examining the *Mesh* domain where the priors are markedly skewed in favor of class *Non-mesh* by 9 : 1. Therefore, according to equation **??**, the only way an example could be classified to class *Mesh* is if[6]

$$\frac{\prod_j O(Mesh|M_{ij}}{\prod_j O(Non-Mesh|M_{ij})} > 9$$

This occurs infrequently with 1 concept description, hence a lot of errors are made on test examples of class *Mesh*. Table 1 shows that as the number of descriptions is increased, it becomes more likely that the ratio of product of LS's for *Mesh* to that for *Non-mesh* exceeds 9, allowing correct classifications on test examples of class *Mesh*.

We now analyze the *Mesh* domain for which the general trend of increased accuracy with greater number of concept descriptions does not hold. To maintain compatability with Dzeroski and Bratko (1992), a single 5-fold cross-validation (CV) trial was used. Unfortunately, this testing methodology is inferior to that of multiple independent trials because only one pass is made over the data in a 5-fold CV trial. This is particularly inappropriate when learning descriptions stochastically, but we maintain this test methodology to be compatible with previous work. This high variance particularly affects the accuracies for 1 stochastically learned concept description and therefore we will restrict our analysis to comparing 5 and 11 descriptions. As the number of descriptions increases from 5 to 11, there is a relatively larger increase in the coverage of rule sets for *Non-mesh* than for *Mesh*. This is to be expected from a data modeling point of view - there are more training

[6]Recall that $O(Class|Rule)$ is instantiated in HYDRA-MM by the LS associated with a rule.

examples of *Non-mesh*. Close examination shows that for 5 descriptions 55.2% of the examples were classified as *Mesh* without contention from rule sets of the other class. 25.0% of examples were classified (without contention) as *Non-mesh*. At 11 descriptions only 4.2% of the examples were classified without contention as *Mesh*, and 25.0% were classified as *Non-mesh*. By only using positive test examples, Dzeroski and Bratko's test methodology violates the assumption that the test and training examples will be drawn from the same distribution. This may explain why increasing the number of descriptions is not beneficial for this domain.

We now present evidence for an undesirable property of the Bayes metric. As the Bayesian metric is more established in the context of learning decision trees, we will illustrate the problem in that context. Consider equation **??** with uniform priors ($\alpha_1 = 1, \alpha_2 = 1, V = 2$). Consider two ways of splitting a tree node covering 11 (p_0) positive and 2 (n_0) negative examples. In the first, 1 positive and 0 negative examples go to the left branch and 10 positive and 2 negative go the right branch (denoted $((1,0),(10,2))$). According to the C-SEP family of measures (Fayyad and Irani, 1992) one should prefer the $((4,0),(7,2))$ over this one. However, the Bayes gain function gives a gain of 5.8×10^{-4} to the first split and 5.5×10^{-4} to the second split. More generally, consider two candidate splits: $((p,0),(p_0-p,n_0))$ and $((p+1,0),(p_0-p-1,n_0))$. It is possible to prove that if $p = \frac{\alpha_2(p_0-1)-n_0\alpha_1}{n_0+2\alpha_2}$ is an integer, then the Bayes gain of these 2 splits will be the same[7]. Even worse, for smaller values of p, as p decreases, the gain will increase. This will lead to larger concept descriptions than is necessary.

[7] To prove this, use equation **??** solve $pr_2(p,0,p_0-p,n_0) = pr_2(p+1,0,p_0-p-1,n_0)$ for p, using the factorial expression for the beta function.

6.2 Multiple rule sets versus multiple rules

Although Kononenko's work on the KRK domain provides a basis for comparing multiple rules against multiple rule sets, his algorithm is disadvantaged by not being able to learn relational rules. To compare multiple rule sets to multiple rules, we adapt HYDRA to produce HYDRA-R which stochastically learns multiple concept descriptions, each consisting of one rule. To encourage learning covers for all disjuncts, a learned rule is only retained if its syntactic form differs from all previously learned rules. If in 20 attempts, a new rule cannot be found, learning for that class and trial is terminated.

For the KRK 320,20 task, HYDRA-MM learning 11 concept descriptions (using *ls-content*) produced an average (over 20 trials) of 221 rules for class *illegal* and 183 for *legal*. We attempted to find 221 syntactically distinct rules using HYDRA-R but on average, could only find 168 rules for *illegal* and 150 rules for *legal* (*MAX-BEST* was set to 20 to increase the chances of finding distinct rules). The resulting average accuracy of 87.9% is statistically lower than that obtained by HYDRA-MM (94.7%) for this task. This suggests that HYDRA-R was unable to effectively learn the minor disjuncts and that if there are minor disjuncts in the concept, it is better to use the multiple rule sets approach rather than the multiple rules approach.

Kononenko's method discourages learning of multiple rules for each disjunct because his Quality metric rewards rules that cover examples not covered by previously learned rules. There is no explicit effort that ensures each disjunct is modeled by more than one rule. Thus, his approach cannot simultaneously satisfy the objectives laid out in section **??** - that all disjuncts should be modeled, and they should be modeled more than once. Although we cannot exclude the possibility that some multiple rule method could learn accurately on this task,

this experiment suggests that at least for this concept, multiple rule sets have an advantage over multiple rules.

6.3 Multiple recursive concept descriptions

Previous work in learning multiple concept descriptions has been restricted to concepts expressible in attribute-value form. The following task requires learning a recursive definition which cannot be simulated using attribute-value rules or decision trees. The document block classification task requires learning some approximation of the following clause:

date-block(X) :- to-the-right(X,Y), date-block(Y)

The extensional definition of *date-block* is used during learning. During testing, the test-time component of HYDRA-MM is called for the recursive literal. In HYDRA-MM, the recursive call has the following semantics. Assume that the clause above is part of the i-th concept description. It returns *true* if the test-time component of HYDRA-MM (using only the i-th concept description) would have classified the example bound to Y as a *date-block*. The recursive call cannot use voting among concept descriptions because that would violate the independence of the concept descriptions. Table 1 shows that use of multiple concept descriptions led to increased accuracy in this domain.

6.4 Comparison to previous work

On the promoters domain, as far as we know, the 92.5% accuracy obtained here is higher than for any symbolic method (ID3 or C4.5) and equal to that of standard neural-net back-propagation (Towell *et al.*, 1990). KBANN (Towell *et al.*, 1990) which can take advantage of prior knowledge gets 96%. The output of KBANN is a set of M-of-N rules which multiple rule sets may be able to approximate.

On the mesh domain, HYDRA-MM with *ls-content* learns rules that cover some negative examples and this explains why its accuracy is much higher than that for GOLEM (29%) reported in Dzeroski and Bratko (1992). However, we believe this effect may be mainly due to a limitation of their testing methodology - testing solely on positive examples.

On the KRK domain, the algorithms of Kononenko *et al.* (1992) can only achieve 82.9% with 1000 examples whereas HYDRA with 100 examples achieves 96.8% primarily because it can learn relational descriptions and because this concept contains small disjuncts for which the multiple rule sets is more appropriate than the multiple rules approach.

7 Conclusions

There are three major results of this work. First, our experiments characterize situations under which learning multiple concept descriptions is most useful; namely, when the hypothesis space is relatively "flat" with respect to the learning gain metric. This opens up the possibility that a learning program could decide how many models to build because it can detect the flatness of the hypothesis space. Second, over several domains our results show that combining evidence from rule sets in a manner consistent with Bayesian theory is a general tool for improving accuracy over that of a single deterministically learned concept description. Finally, we show that for some domains, multiple rule sets have an advantage over multiple rules in that they allow each disjunct to be modeled several times and they also allow learning for all disjuncts in the target concept. Further research topics in this area include determining how many concept descriptions to learn and ways to increase comprehensibility of multiple concept descriptions.

Acknowledgements- We would like to acknowledge Wray Buntine for helpful comments on the Bayesian gain metric and Saso Dzeroski for providing the finite element mesh data. Thanks also to Donato Malerba and Gianni Semeraro for the document block classification domain.

References

Ali K. and Pazzani M. (1992). *Reducing the small disjuncts problem by learning probabilistic concept descriptions* (Technical Report ICS-92-111). Irvine CA: University of California at Irvine, Department of Information & Computer Sciences. To appear in Petsche T., Judd S. and Hanson S. (ed.s), Computational Learning Theory and Natural Learning Systems, Vol. 3. Cambridge, Massachusetts. MIT Press.

Ali K. and Pazzani M. (1993). HYDRA: A Noise-tolerant Relational Concept Learning Algorithm. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.* Chambery, France: Morgan Kaufmann.

Buntine W. (1990). *A Theory of Learning Classification Rules.* Doctoral dissertation. School of Computing Science, University of Technology, Sydney, Australia.

Dolsak B. and Muggleton S. (1991). The application of inductive logic programming to finite element mesh design. In *Proceedings of the Eighth International Workshop on Machine Learning.* Evanston, IL: Morgan Kaufmann.

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

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

Esposito F., Malerba D., Semeraro G. and Pazzani M. (1993). A machine learning approach to document understanding. In *Proceedings of the Second international workshop on multistrategy learning.* Harpers Ferry, WV.

Fayyad U. and Irani K. (1992). The Attribute Selection Problem in Decision Tree Generation. In AAAI-92: Proceedings, Tenth National Conference on Artificial Intelligence, Menlo Park: AAAI Press.

Gams M. (1989). New Measurements Highlight the Importance of Redundant Knowledge. In *European Working Session on Learning* (4th : 1989 : Montpeiller, France). Pitman.

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.

Kononenko I. and Kovacic M. (1992). Learning as Optimization: Stochastic Generation of Multiple Knowledge. In *Machine Learning: Proceedings of the Ninth International Workshop.* Aberdeen, Scotland. Morgan Kaufmann.

Kwok S. and Carter C. (1990). Multiple decision trees. *Uncertainty in Artificial Intelligence, 4*, 327-335.

Lloyd J.W. (1984). *Foundations of Logic Programming.* Springer-Verlag.

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

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

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

Pazzani M. and Brunk C. (1991). Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition, 3*, 157-173.

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

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

Smyth P. and Goodman R. (1992). Rule Induction Using Information Theory. In G. Piatetsky-Shapiro (ed.) *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press, MIT Press.

Towell G., Shavlik J and Noordewier M. (1990). Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90).* Boston, MA. AAAI.