HYDRA-MM: Learning Multiple Descriptions to Improve Classification Accuracy

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

Abstract

For learning tasks with few examples, greater classification accuracy can be achieved by learning several concept descriptions for each class in the data and producing a classification that combines evidence from multiple descriptions. Stochastic (randomized) search can be used to generate many concept descriptions for each class. Here we use a tractable approximation to the optimal Bayesian method for combining evidence from multiple descriptions. Learning multiple descriptions is very useful when additional data is difficult to obtain. The primary result of this paper is that multiple concept descriptions are particularly helpful for improving accuracy in hypothesis spaces in which there are many equally good rules to learn. Another result is experimental evidence that learning multiple rule *sets* yields more accurate classifications than learning multiple rules for concepts containing many disjuncts.

1 Introduction

The general multi-class learning task takes as input a set of concepts (classes) that partition the data, samples of exemplars of each concept and a set of "background concepts." Its output is a description for each class in terms of the background concepts. The descriptions are then used to classify a previously unseen example to one of the concepts. We present a method for learning multiple concept descriptions (each concept description being a rule set) for each concept in the learning task and for combining evidence from these concept descriptions. Evidence is combined from all the descriptions for a given class to produce a final number that signifies how strongly the evidence indicates that the test example should be assigned to that class.

Our experiments show that the resulting classifications are more accurate than those made by the single concept description that is learned from the same data using greedy hill-climbing search. We present results on the following relational problems: predicting finite-element mesh granularity ([8]), learning the concept of "illegality" in the King-Rook-King domain (KRK, [14]), deciding whether a part of an optically-scanned document contains the date of the document ([9]) and predicting whether a person is required to make payments on their student loan ([16]). Although previous work has shown that learning multiple concept descriptions increases accuracy for some kinds of concept descriptions (e.g. decision trees, [5]) there has been no prior work in learning multiple concept descriptions where each description is a rule set or in learning multiple concept descriptions for *relational* concepts. A concept is called a relational concept with respect to some set of relations if it can be succintly described using those relations.

In this paper we present HYDRA-MM which extends the relational learning algorithm HYDRA ([1, 2, 3]) to learn more than one model (figure 1). A concept description is a set of rules that all conclude for the same class. A model is a set of concept descriptions, one for each class in the data. Learning multiple models implies learning multiple concept descriptions for each class. Learning multiple concept descriptions and combining evidence is one way to improve accuracy on "real world" concepts which may not be accurately representable by a set of necessary and sufficient rules but which may need evidence combination from many sources. Rules that are not completely sufficient are modeled in HYDRA by attaching a *degree of logical sufficiency* (LS, [7]) to each rule.

The primary goal of this research is to characterize the conditions under which learning multiple descriptions is beneficial to accuracy. Another goal is to use Bayesian probability theory which postulates that for optimal accuracy, one should make classifications based on votes from *all* hypotheses in the hypothesis space, each vote weighted by the posterior probability of the hypothesis. In practice, however, it is only possible to use a small set of descriptions (hypotheses) so we aim to find the N most probable descriptions. The third goal of this research is to show that using multiple rule *sets* is a better approach than using multiple rules (figure 2, [12]). These approaches differ in that the multiple rules approach tries to model each class with a single, conjunctive rule. However, there are many classes that cannot be accurately modeled with just a single rule with respect to the given set of background concepts.

There has been no previous work in learning multiple rule sets for each class. Because many concepts cannot be accurately modeled using just a single rule, this suggests that the rule sets approach should be able to obtain higher accuracies on such concepts and it should be able to do equally well on other concepts. The

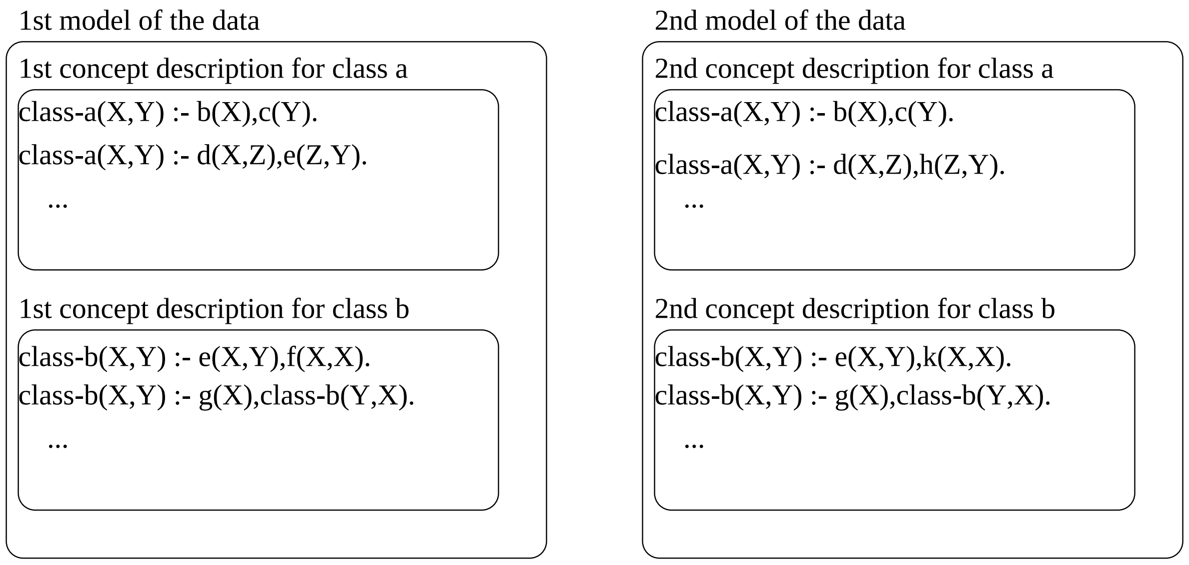
1st model of the data
1st concept description for class a
class-a(X,Y) :- b(X),c(Y).
class-a(X,Y) :- d(X,Z),e(Z,Y).
...
1st concept description for class b
class-b(X,Y) :- e(X,Y),f(X,X).
class-b(X,Y) :- g(X),class-b(Y,X).
...
2nd model of the data
2nd concept description for class a
class-a(X,Y) :- b(X),c(Y).
class-a(X,Y) :- d(X,Z),h(Z,Y).
...
2nd concept description for class b
class-b(X,Y) :- e(X,Y),k(X,X).
class-b(X,Y) :- g(X),class-b(Y,X).
...

Figure 1. Multiple rule sets: most algorithms learn one concept description for each class. HYDRA-MM learns many descriptions for each class and combines evidence from the descriptions using weighted voting.

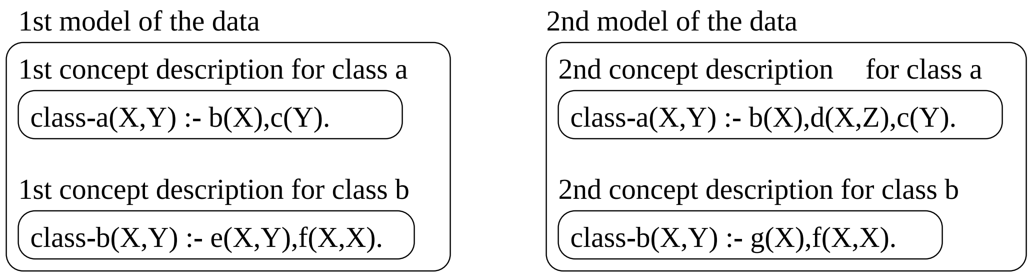
1st model of the data
1st concept description for class a
class-a(X,Y) :- b(X),c(Y).
1st concept description for class b
class-b(X,Y) :- e(X,Y),f(X,X).
2nd model of the data
2nd concept description for class a
class-a(X,Y) :- b(X),d(X,Z),c(Y).
2nd concept description for class b
class-b(X,Y) :- g(X),f(X,X).

Figure 2. The multiple rules approach learns multiple descriptions for each concept. However, because each description consists of a single rule rather than a rule set, it cannot easily model concepts containing multiple disjuncts (subconcepts).

experiments presented in section 7.2 bear this out. Concepts that can only be accurately modeled using more than one rule are said to have multiple disjuncts (subconcepts) with respect to the given set of background concepts. More precisely, a concept is said to contain multiple disjuncts, if there is no purely conjuctive rule[1] for that concept with respect to the given set of variables and hypothesis language that is consistent with all examples and non-examples of that concept.

The next section describes what relational descriptions are and why one would want to learn such descriptions. Section 3 describes the main issues in learning and using multiple descriptions. Section 4 presents the theoretical background for multiple descriptions. Following that, the basic learning strategy used in this work (the separate and conquer strategy) is presented. Section 6 presents HYDRA and section 7 presents the evaluation of our approach.

2 Relational descriptions

Many concepts do not have easily learnable (i.e. succint) attribute-value descriptions but have easily learnable relational descriptions. Attribute-value rules consist of literals (e.g. `>(Age,50)`) which may only compare a variable (e.g. `Age`) against a value (e.g. `50`). Variable-variable comparisons are not allowed. The following is an example of an attribute-value rule (an identifier that starts with an upper-case letter denotes a variable):

$$\text{breast-cancer(Age,...,Degree-of-Malignancy)} \leftarrow \begin{array}{l} >\text{(Age,50)}, \\ >\text{(Degree-of-Malignancy,3)} \end{array}$$

This rule concludes that a woman with some set of values for the variables `(Age,...,Degree-of-Malignancy)` has breast cancer if she is older than 50 years of age and the degree of malignancy is greater than 3 units. Procedurally, the rule takes as input an example which is a sequence of values for the variables and checks to see if those values satisfy all the conditions. If so, we say that the rule *covers* or *matches* the example and that the example *satisfies* the rule. To clarify the terminology used earlier, the learning task here is to classify examples to one of the two classes `(breast-cancer, not-breast-cancer)` and `>` is an example of a background concept. Because it is clear that only one entity is involved in an attribute-value rule, sometimes such a rule is written in the following form:

$$\text{breast-cancer} \leftarrow \text{Age} > 50, \text{Degree-of-Malignancy} > 3.$$

By contrast, relational rules can involve measurements of more than one entity:

[1]An example of a non-conjunctive rule is `class-a` $\leftarrow$ `d,(b or c)`.

breast-cancer(W1) $\leftarrow$ age(W1,Age), >(Age,50), mother(W1,W2), breast-cancer(W2).

This relational rule concludes that a woman has breast cancer if she is older than 50 years of age and if her mother has breast cancer. It uses the 2-arity relations `age`, > and `mother`, and the 1-arity relation `breast-cancer`. This rule is called a recursive rule because a concept (`breast-cancer`) appears in the conclusion and the conditions of the rule.

The general relational learning task is defined as follows. The learning task takes as input (1) a collection of examples belonging to a set of specified classes (e.g. `breast-cancer`, `not-breast-cancer`) which *partition* the example space and (2) a set of background relations or background concepts (e.g. `mother(_,_)`) for which full extensional definitions are provided to the learning algorithm. An extensional definition is a set of all sequences of length two of symbols for which the relation `mother` is true. For example, `(jim,mary)` would be in the extensional definition of `mother` if `mary` is the mother of `jim`. The learning task then is to build a concept description for each class using combinations of the background relations (figure 1). An example is a sequence of ground terms (such as (1,foo)) that is intended to match against the head of a rule. A rule 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)`). Classification of a novel test example proceeds as follows: an attempt is made to match the example with each rule for every class. Hopefully, only rules of one class will match with the example and so it will be classified to that class. Section 6 discusses what to do if it matches with rules of more than one class or if it does not match with any rules of any class.

3 Issues in learning multiple descriptions

Three major decisions that need to be made when using multiple descriptions are the choice of type of model, the method for generating multiple models from the same data set and the method for combining evidence from the descriptions. First, we consider whether to learn multiple rules or multiple rule sets. Two objectives motivate our work on learning multiple rule sets. First, to ensure that more than one attempt is made to learn a rule for each disjunct of each class. Second, to ensure that each minor disjunct (a disjunct that may correspond to a small fraction of the examples of a class) is modeled by a rule. Figure 3 illustrates a concept containing a major disjunct (large dark circle) and a minor disjunct (small dark circle). Light lines indicate the coverage of 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, [18]) which learns an approximation

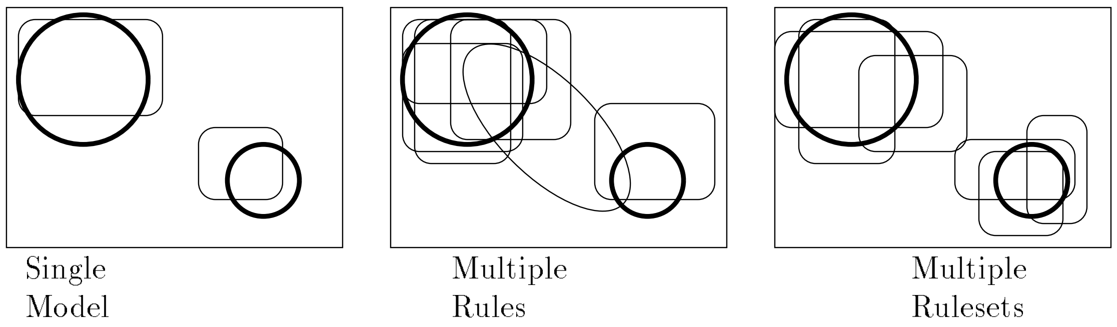

Figure 3. Comparison of three algorithms on a domain where the first class (area inside dark circles) consists of two disjuncts (dark circles). The area outside the dark circles corresponds to the other class. Light lines show coverage of rules learned by the three algorithms.

(rule) for the first disjunct and then removes from the training set the examples covered by that rule in order to learn subsequent rules. In the separate and conquer approach each rule attempts to model one disjunct in the concept. The middle figure illustrates what may be learned by a multiple rules approach in which each rule tries to model an entire concept (both disjuncts). The figure illustrates that because the approach is trying to cover both disjuncts with one rule, and because this cannot be done well in terms of the given set of background concepts, the result is that the learned rule is overly general and covers area outside the first class. Thus, it would have the undesirable consequence of matching test examples of the second class. Finally, the rightmost figure illustrates that HYDRA-MM applies the separate and conquer strategy many times, learning more than one approximation for each disjunct. Thus, HYDRA-MM meets both the criterion for multiple approximations and the criterion for modeling minor disjuncts.

Next, we consider the issue of choosing a method to generate multiple descriptions for the same data set. Previous methods for generating multiple concept descriptions include beam search ([19, 5]), user-intervention ([13]), n-fold cross-validation ([11]) and stochastic search ([12]). The beam search approach works by collecting the N best rules as ranked by some measure such as information gain ([18]). Because this is a multiple rules approach it suffers from the shortcomings we have already mentioned. The user-intervention approach works by allowing the user to examine the best candidate decision nodes for learning a decision tree and then letting the user decide which candidates should be used to learn distinct trees. The disadvantage with this approach is that most users can only be consulted a few times. The third approach, n-fold cross-validation, partitions the training set into n equal subsets and then uses leave- one-out on the subsets to generate n rule sets. In the leave-one-out approach, the i-th subset is excluded from the training set when learning the i-th rule set for a concept. Gams ([11]) uses one version of this method in which a final pass is made which learns using all the data and in which rules are only considered if they appeared in some large number of the n rule sets learned earlier. This approach has the disadvantage that the output is a single model rather than a set of models and most of the research on learning multiple descriptions has shown that keeping multiple models is a better idea than over-committing to one model. The last method for generating multiple descriptions, stochastic search, involves modifying greedy search in that instead of always selecting the best path, the best MAX-BEST paths are retained and a weighted random choice from that set of paths is made. This is the method used in this paper. Its disadvantage is that it requires an educated guess about the value of MAX-BEST but its advantage is that it generates descriptions such that the final classifications are more accurate than those made by combining evidence from descriptions learned by n-fold cross-validation ([4]).

Lastly, we consider the issue of evidence from descriptions. In this paper we will compare two evidence combination methods: the odds[2] form of Bayes rule and voting according to the posterior probability of the model given the training data. Briefly, we will use the odds form of Bayes rule because it is consistent with degree of logical sufficiency (LS, [7]), the type of uncertainty measure used in HYDRA. The degree of logical sufficiency for a class C is also called an odds multiplier because it is multiplied by the prior odds of C to produce the posterior odds of C. The example is classified to the class with the greatest posterior odds. Section 6.2 explains this in more detail.

4 Theoretical background

According to the Bayesian approach to utility maximization, to maximize some utility function u that depends on some action A and an incompletely known state of the world H we should take the action that maximizes the Bayesian expected utility where the expectation is summed over all possible worlds H. For classification, 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 compute this expectation over a small set of highly probable hypotheses. Using the notation in Buntine ([5]), let c be a class, $\mathcal{T}$ be the set of hypotheses the learning algorithm has produced, x be a test example and $\vec{x}$ be the training examples. Then, we should assign x to c

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

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

This means that we must estimate the posterior probability $p(T|\vec{x})$ of each learned model and we must estimate the accuracy $p(c|x,T)$ of each model. In this work, to compute $p(c|x,T)$ we consider the subset of rules in the rule set for class i that have been satisfied by x. The accuracy of the most accurate rule in that subset is used for $p(c|x,T)$.

The other term in equation 1, the posterior probability of the tree, $p(T|\vec{x})$, can be calculated using:

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

where $p(T)$ is the prior of the tree, B is the Beta function,[3] 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.

Besides its use for classification, Buntine has observed that the posterior probability can also be used as a gain metric during learning. The metric is used during learning by choosing to add the decision node to a tree such that the posterior probability of the new tree is maximized. For learning a binary tree from two classes, the following instantiation of equation 2 defines a gain metric:

$$pr_2(n_{11},n_{21},n_{12},n_{22}) = p(T) \times \frac{B(n_{11}+\alpha_1, n_{21}+\alpha_2)}{B(\alpha_1,\alpha_2)} \times \frac{B(n_{12}+\alpha_1, n_{22}+\alpha_2)}{B(\alpha_1,\alpha_2)} \tag{3}$$

where n_{11} and n_{21} respectively denote the numbers of positive and negative examples in the left branch of the node and n_{12} and n_{22} denote the numbers for the right branch. $p(T)$ denotes the prior probability of the tree resulting from addition of the decision node. This gain metric is called the Bayes metric. Ideally, one would want to learn the n most probable concept descriptions with their probability being evaluated globally, but instead we settle for the results of a greedy search.

Now we convert equation 2 for use in classification with rule sets. A rule set can be viewed as a one-sided binary decision tree with compound tests at the nodes, each test corresponding to the body of one rule. Different orderings of rules correspond to different trees but all such trees will make identical classifications. Letting $n_{1,ij}$ and $n_{2,ij}$ respectively denote the number of positive and negative training examples covered by the j-th rule of the i-th class, and V the *set* of rules in the model, we can

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

use equation 2 to compute the posterior probability $p(M|\vec{x})$ of a model M learned by HYDRA:

$$p(M|\vec{x}) \propto p(M) \times \prod_{ij \in V} \frac{B(n_{1,ij} + \alpha_1, n_{2,ij} + \alpha_2)}{B(\alpha_1, \alpha_2)} \tag{4}$$

Now consider using this theory for *learning* rule sets. Using the analogy between rule sets and decision trees, adding a condition to a rule is analogous to adding a condition to the compound test at a decision node. Therefore, change in the pr_2 (equation 3) measures the gain in posterior probability as a result of adding the condition. The problem with using pr_2 directly for learning rules is that pr_2 is symmetric so that a rule 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 use 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.

5 Separate and Conquer Strategy

HYDRA uses a separate and conquer control strategy (table 1) based on FOIL ([18]). In the separate and conquer strategy, rules are learned one at a time. Training examples covered by a rule are removed from the training set and subsequent rules are learned to cover the remaining examples.

A rule for a given class such as `class-a(V1,V2)` is learned by a greedy search strategy. The strategy starts with an empty rule body which covers all remaining positive and negative examples. Next, the strategy considers all literals that it can add to the rule body and ranks each by the information gained ([18]) by adding that literal. The information gain measure favors the literal whose addition to the rule body would cover many positive examples and exclude many negative examples. Quinlan ([18]) defines the information content of a rule covering p_0 positive and n_0 negative examples as:

$$I(p_0, n_0) = -\log_2 \tfrac{p_0}{p_0 + n_0}$$

and the information gained by adding a literal to a rule body such that now the rule covers $p_1 (\leq p_0)$ positive and $n_1 (\leq n_0)$ negative examples as

$$p_1 \times (I(p_0, n_0) - I(p_1, n_1))$$

The strategy keeps adding literals to exclude negative examples until the rule covers no negative examples or there is no candidate literal with positive information gain

Table 1: Pseudo-code for FOIL.

```
FOIL(POS,NEG,Metric,Concept):
 Let POS be the positive examples.
 Let NEG be the negative examples.
 Separate: (begin a new rule)
 Until POS is empty do:
  Let NewRule be the output of Build-rule(POS,NEG,Metric,Concept)
  Remove from POS all positive examples that satisfy NewRule.
 End FOIL

--------------------------------------------
Build-rule(POS,NEG,Metric,Concept)
  Set NewRule to ``Concept if TRUE'' (this rule true for all POS and NEG)
  Until NEG is empty do:
   Conquer: (build a rule body)
    Choose a literal L using Metric
    Conjoin L to body of NewRule.
    Remove from NEG examples that dont satisfy NewRule.
  Return NewRule
  End Build-rule
```

(the latter condition can occur in noisy data sets). Examples covered by the rule are removed from the training set and the process continues to learn on the remaining examples, terminating when no more positive examples are left.

6 HYDRA

HYDRA is derived from FOCL ([17]) which added to FOIL the ability to learn using background knowledge defined in terms of partially correct rules. 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. In [2] we show that this allows HYDRA to learn more accurate descriptions from noisy data.

2. HYDRA attaches the degree of logical sufficiency (LS - a measure of classification reliability) to each rule.

3. The metric used by HYDRA (ls-content) to rank 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 noisy data.

Knowledge Representation- Rules learned by HYDRA are accompanied by a degree of logical sufficiency (our choice for measuring classification reliability). The degree of logical sufficiency of the *j-th* rule of the rule set for the i-th class is defined as follows:

$$ls_{ij} = \frac{p(rule_{ij}(\tau) = true \mid \tau \in Class_i)}{p(rule_{ij}(\tau) = true \mid \tau \notin Class_i)} \tag{5}$$

where τ represents an arbitrary example. The degree of logical sufficiency is a generalization of the notion of sufficiency that the body of a rule has for the head of that rule. Muggleton *et al.* ([15]) show that training coverage is a more reliable indicator of the true accuracy of a rule than apparent accuracy measured from the training data. Accordingly, we choose to use LS as a reliability measure because it has the flavor of measuring both coverage and accuracy.

In understanding how HYDRA classifies a test example, the notion of *representative rule* is useful. The representative rule of a rule set R and a test example x is the rule with the highest reliability chosen from the subset of R that were satisfied by x. A representative rule is selected for each class for which x satisfied at least one rule. The test example is classified to the class whose representative rule has the highest reliability. When a test example satisfies no rule of any class, HYDRA guesses the most frequent class. In the representative rule approach, if more than one rule from a rule set is satisfied, that is not taken as greater evidence that the test example belongs to this class. Our experiments have shown ([4]) that this approach works well when the target description is a compact DNF form with respect to the given set of background concepts. The target description of a concept (with respect to some given set of background concepts) is defined as the shortest consistent description for that concept in terms of the given set of background concepts.

Learning in HYDRA- HYDRA uses the same separate and conquer strategy used in FOCL. However, HYDRA uses the ls-content metric to rank literals. `ls-content` is defined as follows. Assume the j-th rule is being learned and it covers p positive training examples and n negative examples and p_j and n_j respectively denote the number of positive and negative examples uncovered by the first $j-1$ learned rules. Then, `ls-content` is defined as the product of the reliability of the rule (LS) and the coverage (p) of the rule:

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

where $\textit{ls}(p, n, p_j, n_j)$ denotes how the ratio of probabilities in the definition of LS (equation 5) are computed from data. Using this metric HYDRA adds the literal whose addition to the rule maximizes gain with respect to `ls-content`. When there are no literals that cause an increase in `ls-content` or if the rule no longer covers any negative examples, HYDRA removes examples covered by the rule from the training set and learns subsequent rules on the remaining examples. HYDRA learns more accurate rules using `ls-content` than it does using information gain ([2]) for noisy data sets because `ls-content` puts a relatively greater emphasis on learning rules that cover more examples and hence overfit noisy data to a lesser degree.

After all the rules have been learned, HYDRA forms an estimate of the degree of logical sufficiency ls_{ij} associated with rule j of concept i. Note that LS is used as a measure during learning and also for classification reliability. After learning, the LS is estimated from the original set of training examples, not just from the examples uncovered by previously learned rules as used by equation 6. We use the Laplace estimate for a probability to compute the ratio of probabilities in the definition of LS. The Laplace estimate for the probability of an event that occurs on f occasions from a sequence of T trials is $\frac{f+1}{T+2}$. Therefore, replacing the probabilities in the definition of LS by their Laplace estimates yields

$$ls_{ij} = ls(p, n, p_0, n_0) \approx \frac{(p+1)/(p_0+2)}{(n+1)/(n_0+2)} \tag{6}$$

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

Normally, we use LS as the reliability measure of a rule if the gain metric being used is `ls-content`. We use the Laplace estimate for the accuracy of a rule as the reliability measure if the the gain metric is the Bayes metric.

6.1 Learning multiple descriptions

Stochastic search is used to generate multiple concept descriptions. During learning a stochastically generated rule instead of choosing the single literal which maximizes the gain, the best MAX-BEST literals as ranked by the gain metric are considered. A literal is probabilistically chosen from this set with probability proportional to its gain. For the experiments described in section 7.1, we set `MAX-BEST` to 2. Even such a small value gives surprisingly good results. The remainder of this section describes two methods for combining evidence from multiple descriptions. The first method uses LS's of rules to compute the posterior odds for a class. This method is used when the rules are learned using the ls-content metric. The second method, combines the accuracies of rules, weighted by the posterior probability of the model. This method is used when the rules were learned using the Bayes gain metric.

6.2 Evidence combination using LS

Classification using multiple concept descriptions learned with the ls-content metric uses the odds form of Bayes rule ([7]) in which the prior odds of the class ($O(Class_i)$) are multiplied by the odds multipliers ($O(Class_i|M_{ij})$) of the concept descriptions for that class to yield the posterior odds for the class ($O(Class_i|M_i)$):

$$O(Class_i|M_i) \propto O(Class_i) \times \prod_j O(Class_i|M_{ij}) \qquad (7)$$

In practice, for the odds multiplier of rule set M_{ij}, we use the LS of the representative rule of M_{ij} for the current test example.

6.3 Evidence combination using posterior probabilities

For our work with multiple concept descriptions learned using the Bayes gain metric, we use the Laplace estimate of the training accuracy of a rule. For a rule covering p positive and n negative training examples, this estimate is $\frac{p+1}{p+n+2}$. This estimate has an advantage over the maximum likelihood estimate $\frac{p}{p+n}$ in that it does not automatically assign an accuracy of 1 to a rule covering say, one positive and no negative training examples. It is highly unlikely that such a rule would have an accuracy of 1 on test examples. Evidence combination from multiple descriptions of a class learned using the Bayes metric then proceeds as follows. If $a_{i,j}$ denotes the

Laplace accuracy of the representative rule of the j-th concept description for class i and $p_{i,j}$ denotes the posterior probability (weight) of the j-th concept description of class i, the overall posterior probability of class i will be

$$\text{Posterior}(i) = a_{i,1} \times p_{i,1} + \ldots a_{i,n} \times p_{i,n} \tag{8}$$

HYDRA-MM will then assign the test example to the class with the highest posterior probability.

7 Experimental Results

The main goal of our experiments is to see whether multiple concept descriptions are most useful in hypothesis spaces in which there are many, equally good rules to learn for the given gain metric. Other goals are to see how well the greedy approximation to posterior probability does in finding probable descriptions and to compare the multiple rule sets and multiple rules approaches. We are also interested in comparing the Bayes and ls-content metrics and seeing if using non-uniform priors has an advantage over using uniform priors.

Table 2 lists results obtained for four algorithms: `Lsc` denotes learning with `ls-content` and using uniform priors of all classes during evidence combination. `Lsc+p` denotes learning with `ls-content` and using non-uniform priors (estimated using frequencies of each class in the training data) during evidence combination. These methods both learn the same set of rules: all classification differences are due to the influence of priors. `Bayes` denotes learning with the Bayes gain metric and voting using the accuracy of the rule and the probability of the model (equation 8). `Accu` is a simplification of `Bayes` in which concept descriptions vote using just the accuracy of a rule which is equivalent to assuming all models have equal posterior probability.

In addition to testing HYDRA-MM on relational problems, we also tested it on the following domains traditionally regarded as "attribute-value domains": breast cancer recurrence and lymph node classification, DNA promoter predicition, and the KRKP chess problem. In the DNA promoters domain we added the relation `y(X)` which is true if the nucleotide `X` is a `t` or a `c`, and the predicate `r(X)` which is true if `X` is an `a` or a `g` where a nucleotide is one of {a,g,t,c}.

Testing Methodology - For the finite-element mesh data set, the examples are derived from five objects. Dzeroski and Bratko ([8]) 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

Table 2: 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.

Domain	Algo-rithm	Det.	1 CD	5 CDs	11 CDs	Sig.	ties
DNA	Lsc	77.4	75.5	89.6	92.5	NA	24.0
	Lsc+p	77.4	74.5	89.6	92.5	NA	
	Bayes	65.1	75.0	85.9	84.8	NA	
	Accu	65.1	73.6	84.0	85.8	NA	
Lym.	Lsc	79.6	75.5	80.1	82.3	80	21.1
	Lsc+p	78.3	76.4	81.1	82.4	95	
	Bayes	79.3	80.5	82.0	82.8	98	
	Accu	79.3	77.7	79.2	79.8	NS	
Canc.	Lsc	67.2	68.1	69.4	70.4	95	9.3
	Lsc+p	70.4	70.7	70.9	71.1	NS	
	Bayes	71.2	71.5	71.1	71.1	NS	
	Accu	71.2	70.9	72.4	72.1	NS	
KRKP	Lsc	94.5	94.1	95.7	95.7	95	18.7
	Lsc+p	94.3	94.0	95.7	95.7	95	
	Bayes	94.6	94.3	94.8	94.9	NS	
	Accu	94.6	94.0	95.6	95.7	95	
Mesh	Lsc	50.7	60.4	62.9	61.2	NA	3.9
	Lsc+p	32.0	22.3	38.5	52.2	NA	
	Bayes	33.5	36.7	49.3	21.2	NA	
	Accu	33.5	43.0	45.9	39.1	NA	
Docu.	Lsc	99.2	99.2	99.6	100	NS	25.2
	Lsc+p	98.4	98.3	99.6	100	NS	
	Bayes	98.3	97.4	99.0	99.5	NS	
	Accu	98.3	97.4	99.0	99.0	NS	
Stud.	Lsc	82.9	83.3	88.7	89.2	99	7.3
	Lsc+p	87.9	87.0	89.0	89.2	80	
	Bayes	86.1	85.4	88.9	90.4	99	
	Accu	86.1	85.0	87.7	89.4	99	
KRK 160	Lsc	90.6	88.6	90.0	89.9	NS	7.2
	Lsc+p	91.3	89.5	91.0	90.3	NS	
	Bayes	92.3	91.6	91.9	92.0	NS	
	Accu	92.3	90.3	90.9	91.6	NS	
KRK 320	Lsc	93.3	92.5	94.4	94.7	90	5.8
	Lsc+p	93.7	92.5	94.7	95.0	80	
	Bayes	95.5	94.9	95.6	95.5	NS	
	Accu	95.5	93.9	94.8	95.2	NS	

explaining some of the results on this domain. Leave-one-out testing was used on the DNA promoters domain and ten-fold cross validation was used on the document block classification domain. For the King-Rook-King data sets, we used twenty independent training and test sets. For the cancer and lymphography data sets, we partitioned the data twenty times, each time using 70% for training and 30% for testing. 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 King-Rook-King domain entries in Table 2, the figure indicates number of training examples used. Twenty percent of the class labels were randomly reset, to simulate a 20% class noise level.

7.1 Discussion

First, we examine the basic question of whether using multiple models to make classifications leads to increased accuracy. In this discussion, a difference in accuracies is considered to exist only when the accuracies are significantly different using the paired, two-tailed t-test for independent trials and the sign test for the DNA domain where the trials are leave-one-out. Table 2 shows that using multiple descriptions leads to increased accuracy for all tasks except the King-Rook-King task with 160 training examples and a 20% class noise level and some Mesh variants. Our hypothesis is that multiple concept descriptions help most in domains where there are many, equally good rules to learn. To measure this during the learning process of adding a literal to a rule, we keep track of the percentage of occasions on which multiple candidates tied for the best gain (column 8 of Table 2). Table 2 demonstrates a positive correlation between domains for which there are many ways to learn good rules and the domains most helped by multiple models. Table 2 indicates that multiple concept descriptions are most helpful in the DNA promoters domain 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 an astonishing 12.2%! The average number of candidate literals that tied for the highest information gain during learning on this domain was 6.2. That means, on average, the deterministic learning algorithm has to make an arbitrary choice among those six or so literals but using the multiple models approach, we can retain more than one of those options.

A natural question to ask is whether the two gain metrics used lead to significantly different accuracies. Table 2 compares a single, deterministically learned description learned using the Bayes metric to one learned using the ls-content metric. The table shows that `Bayes` does significantly better on four domains, significantly worse on two and there is no difference on three domains. Later in this section we show that the Bayes metric has an undesirable property which may explain its relatively poor performance on those two domains. At eleven descriptions, `Bayes` does better than `Lsc` on only the King-Rook-King 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. The DNA accuracy scores are produced by leave-one-out testing which is known to have high variance. This variance is especially pronounced when there are many ways to learn equally good rules. These factors may explain the difference between `Lsc` and `Bayes` on the DNA domain. The higher accuracy obtained by `Lsc` for the Mesh domain is an artificat because all the test examples are from class `Mesh` and `Lsc` is not penalized for learning overly general rules. The rules learned by `Lsc` for the Mesh concept covered 33% of the training examples of the Non-mesh concept!

Now we examine whether using non-uniform priors helps. For learning a single concept description, priors help significantly for the breast-cancer domain and hurt for the Mesh and Students domains. At eleven concept descriptions, 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 7, the only way an example could be classified to class `Mesh` is if

$$\frac{\prod_j O(Mesh|M_{ij})}{\prod_j O(\text{NonMesh}|M_{ij})} > 9$$

This occurs infrequently with one concept description, hence a lot of test examples of `Mesh` are classified as `Non-Mesh`. Table 2 shows that as the number of descriptions is increased, it becomes more likely that the ratio above will exceed 9, allowing correct classifications on examples of class `Mesh`. In summary, using non-uniform priors always helps or has no effect. It has its greatest effect when the number of models is low.

Another issue to examine is whether weighting models by their posterior probabilities has an advantage over using uniform probabilities. According to equation 8, if uniform probabilities are used, the process of summing evidence for class i is equivalent to adding up the Laplace accuracy estimates for representative rules for class i. If non-uniform probabilites are used, however, the summation becomes a weighted sum. Table 2 shows that at eleven concept descriptions, there is no significant advantage to weighting the descriptions by their probabilities except for the lymphography domain. This suggests that weighting models does not yield a significant advantage given the stochastic method for generating models.

We now analyze the Mesh domain for which learning multiple descriptions does not seem to help. To maintain compatability with Dzeroski and Bratko ([8]), a single five-fold cross-validation (CV) trial was used for testing. Unfortunately, this testing methodology is inferior to that of multiple independent trials because only one pass is made over the data and this leads to high variance. This is particularly inappropriate when learning descriptions stochastically, but we maintain this test methodology to allow comparability with previous work. This high variance particularly affects the accuracies for one stochastically learned description and therefore we will restrict our analysis to comparing five and eleven descriptions. As the number of descriptions increases from five to eleven, there is a relatively larger increase in the coverage of rule sets for `Non-mesh` than for `Mesh`. Accordingly the accuracy on `Mesh` does not increase by much. This is to be expected from a data modeling point of view. There are more training examples of `Non-mesh`. therefore to optimize accuracy, one should become better at classifying examples of `Non-mesh`. Closer examination shows that at five descriptions, 55.2% of the `Mesh` examples were classified as `Mesh` without contention from rules of the other class. Twenty-five percent of the `Non-Mesh` examples were classified without contention as `Non-mesh`. At eleven descriptions, however, only 4.2% of the `Mesh` examples were classified without contention as `Mesh`. The percentage of `Non-mesh` examples classified as `Non-mesh` without contention remained at 25.0%. 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 does not appear to be beneficial for this domain.

We now present evidence for an undesirable property of the Bayes metric. As the Bayes metric is more established in the context of learning decision trees ([5]), we will illustrate the problem in that context. Consider equation 3 with uniform priors and $\alpha_1 = 1$, $\alpha_2 = 1$, $V = 2$. Consider two ways of splitting a tree node covering 11 (p_0) positive and 2 (n_0) negative examples. In the first way, 1 positive and 0 negative examples go to the left branch and 10 positive and 2 negative go the right branch (denoted $((1,0),(10,2))$). Consider another split in which 4 positive and 0 negative examples go to the left branch and the remainder (7 positive and 2 negative) go to the right branch. According to the C-SEP family of measures ([10]) one should prefer the $((4,0),(7,2))$ split over the $((1,0),(10,2))$ split because the first split isolates a greater number of examples of a class. 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[4]. Even worse, for values of p smaller than this critical value, as the value p decreases, the difference in

[4]To prove this, use equation 3 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.

gain between $((p,0),(p_0-p,n_0))$ and $((p+1,0),(p_0-p-1,n_0))$ will get larger in favor of $((p,0),(p_0-p,n_0))$. Because this behavior favors breaking off fewer examples of one class, it will lead to more complex and less accurate concept descriptions.

7.2 Multiple rule sets versus multiple rules

Although the multiple rules approach of Kononenko *et al.* ([12]) on the relational concept of "illegality" in the King-Rook-King domain provides a basis for comparing the multiple rules and multiple rule sets approaches, their algorithm is disadvantaged by not being able to learn relational rules. For instance, they only obtain an accuracy of 82.9% with one thousand training examples whereas HYDRA with just one hundred examples obtains 96.8%. Given the standard set of background concepts for learning the concept of "illegality" in this domain the target description contains rules that are relational and that have no good attribute-value approximation. To provide a fairer comparison of multiple rule sets to multiple rules, we adapt HYDRA to produce HYDRA-R which stochastically learns multiple concept descriptions, each description consisting of one rule. To encourage learning rules for all disjuncts, a learned rule is only retained if its syntactic form differs from all previously learned rules. If in twenty attempts, a new rule cannot be found, learning for that class is concluded.

For the task of learning "illegality" from 320 examples and counter-examples corrupted with 20% class noise, HYDRA-R obtained an accuracy of 87.9% which is statistically lower than the 94.7% obtained by HYDRA-MM. Because HYDRA-R learns rules rather than rule sets, we decided to let HYDRA-R learn $11 \times \overline{r}$ rules in order to compete with HYDRA-MM with eleven rule sets, where $\overline{r}$ denotes the average number of rules in a rule set produced by HYDRA during deterministic learning. The large difference in accuracy between HYDRA-R and HYDRA-MM 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.

7.3 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 rule which cannot be approximated using attribute-value rules or decision trees using the given set of background concepts. The document block classification task ([9]) takes as input a variable corresponding to a region (block) of an optically-scanned document and determines whether that region contains the date of the document. This task requires learning some approximation of the following rule:

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

The extensional definition of `date-block` is used during learning. During testing, in order to determine whether the literal `date-block(Y)` has the truth value `true`, the extensional definition is not used. Rather, the learned description for `date-block` is used. In HYDRA-MM, the recursive call has the following semantics. Assume that the rule above is part of the i-th concept description. It returns `true` if 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 as required for using the evidence combination method (equation 1). Table 2 shows that classifications made by combining evidence from multiple descriptions of a recursive concept are more accurate than those made by a single description.

7.4 Comparison to previous work

On the DNA promoters task, 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 backpropagation ([20]). KBANN ([20]) which can take advantage of prior knowledge (in the form of a set of partially correct rules) gets 96%.

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 ([8]). However, we believe this effect is due to a limitation of testing solely on examples of one class.

On the King-Rook-King domain, the multiple rules approach of Kononenko *et al.* ([12]) 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 approach is more appropriate.

8 Conclusions

There are three major results of this work. First, our experiments show that learning multiple descriptions is most useful when there are many, equally good rules to learn for the classes in the data. This is measured by the percentage of occasions that more than one candidate has the highest gain. This opens up the possibility that a learning program could decide how many descriptions to build as a function of the number of tied candidates. The second result is 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. Learning multiple descriptions can be used as a general tool to boost accuracy, by adding randomization to an existing learning algorithm. The results also show that increasing the number of descriptions leads to increased accuracy. 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 all disjuncts to be modeled at least once. Future directions in this area include ways to determine how many descriptions to learn and ways to increase the comprehensibility of multiple concept descriptions.

Acknowledgements- We would like to acknowledge Wray Buntine, Cliff Brunk, Saso Dzeroski, Donato Malerba and Gianni Semeraro for help in this project.

References

[1] K. Ali and M. Pazzani, "Reducing the small disjuncts problem by learning probabilistic concept descriptions," in T. Petsche, S. Judd and S. Hanson (ed.s), *Computational Learning Theory and Natural Learning Systems*, Vol. 3, MIT Press, 1994.

[2] K. Ali and M. Pazzani, "HYDRA: A Noise-tolerant Relational Concept Learning Algorithm," *Proc. 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Press, Chambery, France, pp. 1064-1070, 1993.

[3] K. Ali and M. Pazzani, "On Learning Multiple Descriptions of a Concept," *Proc. on Sixth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, pp. 476-483. 1994.

[4] K. Ali and M. Pazzani, "Error Reduction through Learning Multiple Descriptions", submitted to *Machine Learning.*

[5] W. Buntine, *A Theory of Learning Classification Rules*, doctoral dissertation, School of Computing Science, University of Technology, Sydney, Australia, 1990.

[6] P. Clark and R. Boswell, "Rule Induction with CN2: Some Recent Improvements," *Proceedings of the European Working Session on Learning*, 1991.

[7] R. Duda, J. Gaschnig and Hart P, "Model design in the Prospector consultant system for mineral exploration", in D. Michie (ed.), *Expert Systems in the Micro-electronic Age,* Edinburgh University Press, 1979.

[8] S. Dzeroski and I. Bratko, "Handling noise in Inductive Logic Programming," *Proc. International Workshop on Inductive Logic Programming*, ICOT, Tokyo, Japan, 1992.

[9] F. Esposito, D. Malerba, G. Semeraro and M. Pazzani M, "A machine learning approach to document understanding," *Proc. 2nd International Workshop on Multistrategy Learning.*, Harpers Ferry, WV., 1993.

[10] U. Fayyad and K. Irani, "The Attribute Selection Problem in Decision Tree Generation," *Proc. 10th National Conference on Artificial Intelligence*, AAAI Press, San Jose, CA., pp. 104-110, 1992.

[11] M. Gams, "New Measurements Highlight the Importance of Redundant Knowledge," *Proc. 4th European Working Session on Learning*, Pitman Press, Montpeiller, France, 1989.

[12] I. Kononenko and M. Kovacic M, "Learning as Optimization: Stochastic Generation of Multiple Knowledge," *Proc. 9th International Workshop on Machine Learning*, Morgan Kaufmann Press, Aberdeen, UK, pp. 257-262, 1992.

[13] S. Kwok and C. Carter, "Multiple decision trees," *Uncertainty in Artificial Intelligence*, Vol. 4, 1990, pp. 327-335, 1990.

[14] S. Muggleton, M. Bain, J. Hayes-Michie and D. Michie, "An experimental comparison of human and machine-learning formalisms," *Proc. 6th International Workshop on Machine Learning*, Morgan Kaufmann Press, Ithaca, NY., pp. 113-118, 1989.

[15] S. Muggleton, A. Srinivasan and Bain M, "Compression, Significance and Accuracy," *Proc. 9th International Workshop on Machine Learning*, Morgan Kaufmann Press, Aberdeen, Scotland, pp. 338-347, 1992.

[16] M. Pazzani and C. Brunk, "Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning," *Knowledge Acquisition*, Vol. 3, 1991, pp. 157-173.

[17] M. Pazzani and D. Kibler, "The utility of knowledge in inductive learning," *Machine Learning*, Vol. 9, No. 1, 1991, pp. 57-94.

[18] R. Quinlan, "Learning logical definitions from relations," *Machine Learning*, Vol. 5, No. 3, 1990.

[19] P. Smyth and R. Goodman, "Rule Induction Using Information Theory," in G. Piatetsky-Shapiro (ed.) *Knowledge Discovery in Databases*, MIT Press, 1992.

[20] G. Towell, J. Shavlik and M. Noordewier, "Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks," *Proc. 8th National Conference on Artificial Intelligence*, AAAI Press, Boston, MA., pp. 861-866, 1990.