1
Classification using Bayes Averaging of Multiple, Relational Rule-based Models

Kamal Ali and Michael Pazzani

Department of Information and Computer Science,
University of California, Irvine, CA, 92717, USA.
ali@ics.uci.edu, pazzani@ics.uci.edu

ABSTRACT We present a way of approximating the posterior probability of a rule-set model that is comprised of a set of class descriptions. Each class description, in turn, consists of a set of relational rules. The ability to compute this posterior and to learn many models from the same training set allows us to approximate the expectation that an example to be classified belongs to some class. The example is assigned to the class maximizing the expectation. By assuming a uniform prior distribution of models, the posterior of the model does not depend on the structure of the model: it only depends on how the training examples are partitioned by the rules of the rule-set model. This uniform distribution assumption allows us to compute the posterior for models containing relational and recursive rules. Our approximation to the posterior probability yields significant improvements in accuracy as measured on four relational data sets and four attribute-value data sets from the UCI repository. We also provide evidence that learning multiple models helps most in data sets in which there are many, apparently equally good rules to learn.

1.1 Introduction

There has been much work in learning relational models of data in recent years (e.g. FOIL: [Quinlan90]; FOCL: [Pazzani-Kibler91]; GOLEM: [Muggleton-Feng90], CLINT: [deRaedt-Bruynooghe88], ML-SMART: [Bergadano-et-al88]). Here we present results that apply Bayesian probability theory (e.g. [Berger85]) to learning and classification using class descriptions consisting of first-order (relational) rules. According to this theory, it is necessary to learn several models of the data. In our work, each model consists of one class description per class in the data so learning multiple models implies learning many descriptions for each class. Although there has been prior work in applying Bayesian probability theory for learning and combining evidence from multiple decision trees ([Buntine90]) there has been no such work for relational rules or for rule-set models. A rule-set model is defined to consist of a set of class descriptions, one description per class in the data. Each description, in turn, consists of a set of rules that all conclude for the same class (Table 1.1).

Our core algorithm is based on FOIL which learns a disjoint covering for a class. That is, it learns rules for a class that cover disjoint subsets of the training examples of that class. Therefore, ideally, each learned rule should correspond to a disjunct or subclass in the class. By applying FOIL stochastically many times for each class, we are able to model each disjunct more than once. This in turn leads to a better overall model of each subclass and hence of the class. Our goal is to demonstrate that classifications obtained

[1] *AI and Statistics V.* Edited by F. Flinstone and B. Bunny. ©1995 Me-Me-Me Publishers.

TABLE 1.1. HYDRA-MM learns many class descriptions for each class. For each class it combines evidence from descriptions of that class to calculate the expectation that a test example belongs to that class. The test example is assigned to the class that maximizes that expectation.

	1st model of the data	2nd model of the data

Description for class a	class-a(X,Y) :- b(X), c(Y).
	class-a(X,Y) :- d(X,Z), e(Z,Y).
		class-a(X,Y) :- b(X), c(Y).
		class-a(X,Y) :- d(X,Z), h(Z,Y).

Description for class b	class-b(X,Y) :- e(X,Y), f(X,X).
	class-b(X,Y) :- g(X), class-b(Y,X).
		class-b(X,Y) :- e(X,Y), k(X,X).
		class-b(X,Y) :- g(X),class-b(Y,X).

by a set (ensemble) of models learned by stochastic application of the FOIL algorithm are more accurate than those produced by a single, deterministic application of the FOIL algorithm.

Previous work in learning multiple models consisting of rules has been done by [Gams89] and [Kononenko92]. The limitation of Gams' approach is that it does not retain the multiple models for classification. Instead, after learning several models of the data, a single model is constructed whose rules are chosen from the rules of the previously learned models. Kononenko & Kovacic ([Kononenko92]) have also done work in learning multiple models but in their work, each class in each model is described using just a single rule. Realizing that many classes cannot be accurately described using just a single rule, they advocate the use of a rule-selection metric that rewards rules that cover (i.e. match) training examples not covered by previously learned models for that class. But then this method does not reward learning more than one description for each disjunct (subclass) in the class.

There are two main empirical results of this paper. The first is the development of an approximation to the posterior probability of a rule-set model and the empirical demonstration that weighting the classifications of models by their posterior probabilities leads to more accurate classifications than those obtained by the model learned from a single, deterministic application of FOIL. The second result is that learning and using multiple models is especially helpful in data sets where there are many, apparently equally good rules to learn. To demonstrate these results, we adapt a relational learning algorithm (HYDRA, [Ali-Pazzani93]) to learn multiple models, yielding HYDRA-MM. HYDRA-MM learns multiple rule-set models, therefore it also learns more than one class description for each class. Because HYDRA is a *relational* learning algorithm, it can learn the more expressive relational rules as well as attribute-value rules.

1.2 Theoretical development

For classification, Bayesian probability theory advocates making the classification whose expected accuracy is maximal, with the expectation being taken over all possible models in the model space of the learning algorithm (this process is called Bayes averaging). In practice, we only compute this expectation over a small set of highly probable models. Using the notation in [Buntine90], let c be a class, $\mathcal{H}$ denote the model space, x be a test example, $\vec{x}$ denote the training examples and $\vec{c}$ denote the class labels of the training examples. Then, we should assign the test example x to the class c with the highest posterior probability:

$$E(p(c|x,\mathcal{H})) = \sum_{h \in \mathcal{H}} p(c|x,h)p(h|\vec{x},\vec{c}) \tag{1.1}$$

Here, $p(h|\vec{x},\vec{c})$ denotes the posterior probability of a model h given the training examples and their class labels. The posterior probability of the class given the test example x and the model h ($p(c|x,h)$) can also be thought of as the degree to which h endorses class c for example x. Equation 1.1 is only true when $\mathcal{H}$ is a finite or enumerably infinite set. If not, one also needs to integrate over all continuous variables ([Buntine90], pg. 99) to obtain the expectation $E(p(c|x,\mathcal{H}))$. In this paper, we will assume that one can get good approximations for the posterior without including the extensions for continuous variables. The good experimental results in section 1.5 suggest that significant accuracy improvements can be obtained despite this assumption.

Now we review how to compute the posterior probability $p(h|\vec{x},\vec{c})$ of a decision tree. This form will be used in section 1.2.1 to develop the approximation for the posterior probability of a rule-set model. Applying Bayes rule to the expression for posterior probability yields:

$$p(h|\vec{x},\vec{c}) = \frac{p(\vec{x},\vec{c}|h)p(h)}{p(\vec{x},\vec{c})} \propto p(\vec{x},\vec{c}|h)p(h) \tag{1.2}$$

The $p(\vec{x},\vec{c})$ term can be ignored because in this paper we are only interested in the *relative* posterior probabilities of the models. The term $p(h)$ refers to the prior probability of the model. The uniform distribution is used if there is no *a priori* reason to favor one model over another. We use the uniform distribution in this paper.

The term $p(\vec{x},\vec{c}|h)$ is called the likelihood of the evidence (i.e. the training examples $\vec{x}$ together with their class labels $\vec{c}$) given the model. It measures how likely h is to produce the evidence $(\vec{x},\vec{c})$. For classification, it is used in calculating how likely a model is to generate a set of training examples with their associated class labels.

In order to apply this theory, a way of calculating the likelihood function, $p(\vec{x},\vec{c}|h)$, from data is needed. In order to do this, Buntine assumes that it is useful to partition the example space into V subsets where each subset should be modeled with its own set of parameters. That is, the assumption is that a good model can be built for each subset. (This is equivalent to assuming that confounding examples of one subset with those from another will cause modeling difficulties.) For classification, we will model each subset with C parameters where C represents the number of classes[2]. The i-th parameter of a subset will denote the probability of generating[3] an example of class i from that subset.

In order to derive a useful form for $p(\vec{x},\vec{c}|h)$, we note that by assuming that the examples in the training set are independently generated, we can write:

$$p(\vec{x},\vec{c}|h) = \prod_{i=1}^{N} p(x_i,c_i|h)$$

where N denotes the number of training examples. Furthermore, if we let $n_{j,k}$ denote the number of training examples of class j from the k-th subset (one of the V subsets), and we let $\phi_{j,k}$ denote the probability of generating a single example of class j in the k-th

[2]Note that each subset may correspond to an entire class or to some subclass of a class.

[3]Equivalently, the i-th parameter will represent the probability of finding an example of class i in the subset.

subset, we can write:

$$p(\vec{x}, \vec{c}|h) = \prod_{k=1}^{V} \prod_{j=1}^{C} \phi_{j,k}^{n_{j,k}} \tag{1.3}$$

One can then show ([Buntine90]) that the contribution to the posterior from the k-th subset can be modeled by:

$$\frac{B_C(n_{1,k} + \alpha, \ldots, n_{C,k} + \alpha)}{B_C(\alpha, \ldots, \alpha)} \tag{1.4}$$

where B_C is the C-dimensional beta function and α is a parameter[4] which denotes the "weight" in number of examples that should be associated with the prior estimate $(1/C)$ of $\phi_{j,k}$. Putting equations 1.3 and 1.4 together, we get the likelihood:

$$p(\vec{x}, \vec{c}|h) = \prod_{k=1}^{V} \frac{B_C(n_{1,k} + \alpha, \ldots, n_{C,k} + \alpha)}{B_C(\alpha, \ldots, \alpha)} \tag{1.5}$$

Note that the postulated subclasses appear as leaves for decision trees. Finally, we get an expression for the posterior probability, $p(h|\vec{x}, \vec{c})$ that can be computed from data:

$$p(h|\vec{x}, \vec{c}) \propto p(h) \times \prod_{k=1}^{V} \frac{B_C(n_{1,k} + \alpha, \ldots, n_{C,k} + \alpha)}{B_C(\alpha, \ldots, \alpha)} \tag{1.6}$$

To summarize the preceding sections: in order to decide which class maximizes

$$E(p(c|x, \mathcal{H})) = \sum_{h \in \mathcal{H}} p(c|x, h) p(h|\vec{x}, \vec{c})$$

one can use equation 1.6 for the term $p(h|\vec{x}, \vec{c})$. For the endorsement term $(p(c|x, h))$, it is preferable to use a Laplace estimate ([Kruskal78], p. 256; $\frac{n_{i,j}+\alpha}{n_{.,j}+C\alpha}$) rather than the maximum likelihood estimate $(\frac{n_{i,j}}{n_{.,j}})$ because the maximum likelihood estimate assigns a value of 1 to a leaf covering just one example of a class and no examples of other classes. For such a leaf, it is not very likely that the true estimate of $p(c|x, h)$ is 1.

1.2.1 Posterior probability of a rule-set model

The learning algorithm presented in this paper learns rule-set models. A single rule-set model is a set of class descriptions, one per class. Each class description, in turn, is a set of rules. This section uses the method of deriving the posterior probability of a decision tree to develop a similar method for a rule-set model.

The derivation of the expression for the posterior probability of a tree only depends on assuming that the example space can be partitioned into V disjoint subsets. For trees, if we assume that each leaf of a learned tree models one of those subsets, we can estimate statistics of that subset from the numbers of training examples covered by that leaf. However, the difference between the tree and the rule-set model is that any example in the example space can only belong to one leaf whereas it is possible that rules may be learned in a way such that an example is covered by more than one rule of a class, or by more than one rule of different classes. To avoid this problem *within* a class description,

[4] Because we will use a single value of α for all the V subsets, α does not need j, k subscript.

we will assume that some ordering is imposed on the rules within each class description. Furthermore, we will add a default rule to the end of each class description. For the i-th class description, the default rule is $Class_i(...) \leftarrow true$. Thus we have the situation where each class description forms a partitioning of the example space. Because each class forms its own partitioning of the example space, the situation where a test example matches rules of more than one class is also not a problem. For decision trees, the posterior is a function of the way in which the example space is partitioned by the leaves of the decision tree. Analogously, for rule-set models, the posterior of each class description is a function of the way in which the example space is partitioned by the rules of that class description. To compute the posterior of the rule-set model then, we just take the geometric average of the posteriors of the class descriptions.[5] One way of interpreting this approach is that each class description provides an estimate of the posterior using a partitioning of the training data and that we are just forming the average of those estimates.

Let C denote the number of classes in the data, M denote a learned rule-set model, R_i denote the set of rules (including the default rule) for class i after some ordering has been imposed on those rules, $n_{1,ij}$ denote the number of training examples of class i covered by rule j of class i and $n_{2,ij}$ denote the number of training examples of classes other than class i covered by rule j of class i. Then, the posterior probability of a rule-set model is the geometric average of the estimates provided by each class description:

$$p(M|\vec{x},\vec{c}) \propto p(M) \times \left(\prod_{i=1}^{k} \prod_{ij \in R_i} \frac{B(n_{1,ij}+\alpha, n_{2,ij}+\alpha)}{B(\alpha,\alpha)} \right)^{1/C} \tag{1.7}$$

Note that we are assuming that if a rule of a class covers a training example of some other class, it does not matter which other class that example belongs to - the loss function for every other class is the same.

In order to use a rule-set model for classification using equation 1.1, one other term must be estimated from data. This is the $p(c|x,h)$ term - the degree to which model h endorses class c for test example x. To compute the endorsement for some class i, we will only consider the rules in the i-th class description of model h. Next, we need to impose an ordering on the rules of the i-th class description. To do this, we define $r_{ij}(x)$ to be a random variable that takes on value `true` if example x satisfied the conditions of rule j (of class i) and takes on value `false` otherwise. The rules of the i-th class description are then ranked in order of most accurate first where the accuracy of a rule is defined as $p(i|r_{ij}(x) = true)$. The training data is used to estimate the accuracy. Let $n_{1,ij}$ denote the number of training examples of class i covered by the j-th rule of class i, and let $n_{2,ij}$ denote the number of training examples of other classes covered by the rule. Then, we use the Laplace estimate of $p(i|r_{ij}(x) = true)$ from the training data:

$$p(i|r_{ij}(x) = true) \approx \frac{n_{1,ij}+1}{n_{1,ij}+n_{2,ij}+2}$$

Finally, the endorsement $p(i|x,h)$ is assigned the accuracy of the first (examined in order of most accurate first) rule in the i-th class description of h that is satisfied by the test example x. Because of the default rule and the ordering imposed on rules of a class

[5]The geometric average of numbers $x_1 \ldots x_n$ is $(\prod_i x_i)^{1/n}$.

description, it is guaranteed that exactly one rule of each class description will match each test example. Such an ordering is needed firstly to produce a *partition* of the training data as required by the framework of [Buntine90] and secondly to avoid the issue of how to combine evidence from rules (called the "rule overlap problem," [**?**]).

The foregoing paragraphs illustrate how to compute the posterior probability of a rule-set model and how to estimate it from data. They also explain how to estimate the degree of endorsement. Now we will consider how to *search* for models with high posterior probabilities.

1.2.2 Bayesian statistics in learning

As Buntine ([Buntine90]) has observed, if our goal is to find the most probable models, posterior probability should be used as a rule-selection (or tree-selection) metric. During greedy hill-climbing for example, one should prefer adding the decision node or condition to the model which maximizes the posterior probability of the model. As we will be concerned with a learning algorithm that learns a rule by successively specializing the rule (i.e. by greedily adding conditions to the body of the rule), we will add the condition $\mathcal{L}$ which maximizes

$$gain = p(h + \mathcal{L}|\vec{x}, \vec{c}) - p(h|\vec{x}, \vec{c}) \tag{1.8}$$

where $h + \mathcal{L}$ denotes the model with the addition of the condition $\mathcal{L}$ to the rule currently being learned. The difference in posterior probability is referred to as the "gain" and this gain metric is called the Bayes gain metric.

This general principle is instantiated for rule-learning as follows. During learning for class i, there are only two "virtual" classes: training examples of class i are called "positive training examples" and examples of other classes are called "negative training examples." The addition of the condition $\mathcal{L}$ to the rule separates out these two virtual classes into two subsets: T: the examples that match the new rule and, F: those that do not. By observing that this situation is analogous to a binary test at an internal node of a decision tree, we obtain the following for the contribution to the posterior probability by $\mathcal{L}$:

$$gain = p(\mathcal{L}) \times \frac{B(p + \alpha_1, n + \alpha_2)}{B(\alpha_1, \alpha_2)} \times \frac{B(P - p + \alpha_1, N - n + \alpha_2)}{B(\alpha_1, \alpha_2)} \tag{1.9}$$

where p and n denote the numbers of positive and negative training examples covered by the rule and P and N denote the numbers of training examples left uncovered by previous rules. We can disregard the prior probability ($p(\mathcal{L})$) of the condition $\mathcal{L}$ because we are assuming the uniform prior distribution for all rule-set models. We make this assumption for parsimony rather than because the theory will not permit other kinds of prior distributions.

Note however that equation 1.9 is symmetric: that is, it will assign the same credit to a condition that excludes as many examples as to a condition that includes those same examples. This poses no problem for decision trees, but when learning rules, we are only interested in modeling the T subset: the subset of examples that are included by the rule. Therefore, we modify the definition of gain in equation 1.8 to be 0 if the ratio of positive examples to negative examples decreases as a result of addition of $\mathcal{L}$.

Now we consider how to test the theory presented in this section using the learning algorithm HYDRA ([Ali-Pazzani93]) that learns relational rules for noisy, multiclass tasks.

1.3 HYDRA: Learning a single rule set model

HYDRA uses a separate and conquer control strategy based on FOIL ([Quinlan90]). FOIL only learns on data sets consisting of two classes, one of which must be identified as the "positive" class. FOIL only learns a description for the positive class. In the "separate and conquer strategy" a rule is learned and then the training examples of the positive class which are covered by that rule are taken out of the training set. Subsequent rules are learned using all the negative examples and the remaining positive examples. Learning terminates when no positive examples are left in the training set. To use this algorithm, the user does not need to specify the number of rules to learn or the length of the rules to be learned. The pseudo-code for FOIL is presented in table 1.2. We adapt FOIL to learn for multiclass tasks (tasks with more than two classes) simply by calling FOIL once for each class in the data. Although FOIL uses the information gain metric ([Quinlan90]) to decide which rule condition to add to the rule next, we will discuss FOIL in the context of using the Bayes gain metric.

FOIL begins to learn a rule for *class-a(X,Y)*, for example, by starting with the rule that has the empty body (the body of this rule is satisfied by any example, positive or negative):

$$\textit{class-a(X,Y)} \leftarrow$$

Then, it ranks each possible condition that it might add to the body of the rule by calculating how much the posterior probability of the model would be increased if the empty body were to be replaced by that condition. FOIL adds the rule condition which yields the largest gain. FOIL continues to add rule conditions until the body of the rule is not true for any negative example. Then, it removes positive examples covered by the rule and learns subsequent rules for the remaining training examples. The process terminates when no positive examples are left in the training set.

HYDRA differs from FOIL in learning a description for each class in the data. Another major difference between HYDRA and FOIL is that HYDRA attaches a reliability measure to each rule. For the experiments in this paper, the training data Laplace estimate of the "accuracy" of the rule (subsection 1.2.1) is used as the rule reliability measure. After learning has finished, the rules within each class description are ordered from most to least accurate.

Classification in HYDRA, which learns a single class description per class, proceeds as follows. If a test example satisfies (non-default) rules of just one class then that class is assigned to the test example. If the test example satisfies (non-default) rules from more than one class, HYDRA chooses the class corresponding to the satisfied rule with the highest reliability. Ali & Pazzani ([Ali-Pazzani93]) show that this modification to FOIL makes the system much more accurate in noisy domains. If the test example does not satisfy any non-default rules of any class, it is assigned to the most frequently occurring class (the Laplace accuracy of the default rule associated with the most frequent class will be higher than the associated accuracies of the other default rules).

The Laplace accuracy of a rule covering p positive and n negative training examples is $(p+1)/(p+n+2)$. The advantage of using this rather than the maximum likelihood estimate $(p/(p+n))$ is that it does not assign an accuracy of 1 to a rule covering just a small number of positive examples and no negative examples. The reliability of such rules is quite

TABLE 1.2. Pseudocode for FOIL: the separate and conquer strategy.

```
FOIL(Positive-Examples,Negative-Examples,Metric):
 Let POS be Positive-Examples.
 Until POS is empty do:
    Let NEG be Negative-Examples.
    Set NewClause to the clause with the empty body
    Until NEG is empty do:
            Conquer: (add a condition to clause body)
            Choose a condition L using Metric
            Conjoin L to body of NewClause.
            Remove from NEG examples that dont satisfy NewClause.
    Separate: Remove from POS all positive examples that
              satisfy NewClause.
```

different from that of a rule covering say a hundred positive and no negative examples. Yet, both would have the same maximum likelihood accuracy estimate. Additionally, among rules covering no negative examples, the Laplace estimate rises monotonically with increasing coverage of positive training examples.

1.4 Learning multiple models with HYDRA

Stochastic (randomized) search is used by HYDRA-MM to find multiple models. Ideally, one would want to find the models which have the highest posterior probability but in practice we use stochastic greedy search with respect to posterior probability to try to find probable models. If there are local maxima in the posterior probability space, we may not find the globally most probable model but we will find models that occupy local maxima.

During the process of deciding which rule condition to add to the body of a rule being learned, HYDRA-MM stores all candidates whose gain is within some factor β of the gain of the best candidate. HYDRA-MM then chooses a rule condition stochastically from this set; the probability of a rule condition being chosen is equal to its contribution to posterior probability. Thus, rule conditions that add more to posterior probability have a correspondingly greater probability of being chosen.

Classification in HYDRA-MM: To compute the expectation of a class given a test example and the set of learned rule models, we use equation 1.1, using the posterior probabilities of rule-set models and their degrees of endorsement. The test example is assigned to the class with the highest expectation. Another way of viewing equation 1.1 is that for a given class c, it collects endorsement evidence from each rule-set model. Endorsements of rule-set models with higher posterior probabilities are given correspondingly greater weight.

1.5 Experimental Results

The main goal of these experiments is to verify that learning multiple rule-set models in a way consistent with Bayesian probability theory leads to improvements in classification

TABLE 1.3. Accuracies obtained by learning multiple models. The numbers in the 3rd-to-last column give the significance level (SIG) at which 11 class descriptions (CDs) accuracy differs from that of 1 deterministic description according to the paired 2-tailed t-test (sign test for DNA). NS - not significant, NA - t-test not applicable. KRK 160,20 denotes learning from 160 training examples with 20% artificial class noise on the King-Rook-King task. β is 0.8.

Domain	Deter-ministic	1 CD	2 CDs	5 CDs	11 CDs	SIG.	ties	# Train eg.s
Promoters	65.1	75.0	77.2	85.9	84.8	98	24.0%	105
Lymphography	79.3	80.5	80.5	82.0	82.8	98	21.1%	99
Cancer	71.2	71.5	71.1	71.1	71.1	NS	9.3%	181
KRKP	94.6	94.3	94.8	94.8	94.9	NS	18.7%	200
Document	98.3	97.4	98.4	99.0	99.5	NA	25.2%	220
Students	86.1	85.4	86.9	88.9	90.4	99	7.3%	100
KRK 160,20	92.3	91.6	91.8	91.9	92.0	NS	7.2%	160
KRK 320,20	95.5	94.9	95.3	95.6	95.5	NS	5.8%	320

accuracy when compared to the accuracy of a single rule-set model learned by deterministic hill-climbing using the same training set. Another goal is to try and understand why learning multiple models leads to large gains in accuracy in some domains (e.g. DNA promoters in table 1.3) but more modest gains in other domains. Our hypothesis is that multiple class descriptions help most in domains where there are many, equally good rules to learn. We measure this by the percentage of attempts at adding a rule condition during learning in which more than one candidate condition equaled the highest posterior probability. That is, the criterion tries to measure how often more than one candidate offers exactly the same increase in posterior probability (a gain tie). In such a situation, the greedy deterministic learner is at a disadvantage because it cannot explore both search paths: it must arbitrarily choose one rule condition.

Table 1.3 presents results on the relational problems of learning the concept of "illegality" in the King Rook King chess data set ([Muggleton89]), deciding whether a part of an optically-scanned document contains the date of the document ([Esposito93]) and predicting whether a person is required to make payments on their student loan ([Pazzani-Brunk91]). We also tested HYDRA-MM on 4 attribute-value problems. The accuracies presented in the table are averages of thirty independent trials (except for the Document domain for which 10-fold cross-validation was used). Table 1.3 indicates that learning multiple descriptions improves accuracy by a statistically significantly margin on three of the domains and increases accuracy by a numerically significant margin on the Document domain.[6] No significant loss was incurred on any domain. The table indicates that using multiple models is most helpful on domains in which there are many gain ties.

Recursive class descriptions: In the document task ([Esposito93]) the problem is to determine whether an example representing a document or a part of a document is a "date-block". A "date-block" is that part of a document which contains the date (other parts being the body, the signature-block etc). The goal is automatic classification (using optical character recognition and a learned, relational rule-base) of documents into types of documents such as letters, orders, etc. For this domain, HYDRA and HYDRA-MM

[6]The t-test cannot be used for this domain as it assumes that the trials are independent which they certainly are not when doing 10-fold cross validation.

learned recursive rules such as:

$$\text{date-block}(X) \leftarrow \text{height-small}(X), \text{to-the-right}(X,Y), \text{date-block}(Y).$$

During learning, the extensional definition of *date-block* is used. That is, the recursive call to *date-block* is satisfied if the value bound to variable Y is a member of the set (the extensional definition) of positive examples for *date-block*. During classification of test examples, to determine if the recursive call is satisfied, we use the set of rules (the intensional description) learned for *date-block*. One issue that arises when recursive descriptions are mixed with multiple models is whether in order to check if the recursive call succeeds one should check all models of *date-block* or just the current model. That is, if the rule being matched against a test example comes from the i-th model, then to determine if a recursive call succeeds, should one check just rules of the i-th model or those of all models? We chose the former option in order to maintain the independence of the models. We also have to ensure that the matching process of a rule set (containing recursive rules) to an example terminates. Currently, we employ an absolute limit on the number of recursive calls to ensure termination. Table 1.3 shows that using multiple recursive models in the Document domain leads to greater accuracy than using a single recursive model.

1.6 Conclusion

There are four contributions of the work presented here. Firstly, we have presented a form for the posterior probability of a rule-set model in accordance with Bayesian probability theory. Secondly, we have provided some empirical evidence that that form leads to improvements in classification accuracy when compared to the accuracy obtained by the single description learned by deterministic greedy search over the same training set. Stochastic hill-climbing was used to learn more than one model from the same training set. Thirdly, we have characterized an experimental property (gain ties) which indicates when learning of multiple models is useful. Finally, we have demonstrated the learning of multiple models containing recursive rules. The resulting classifications were more accurate than those made by the single recursive description learned through deterministic hill-climbing.

1.7 References

[Ali-Pazzani93] 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.

[Bergadano-et-al88] Bergadano F., Giordana A. (1988) A Knowledge Intensive Approach to Concept Induction. In *Proceedings of the Fifth International Conference on Machine Learning.*, Ann Arbor, MA: Morgan Kaufmann.

[Berger85] Berger J. O. (1985). *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York.

[Buntine90] Buntine W. (1990). *A Theory of Learning Classification Rules.* Doctoral dissertation. School of Computing Science, University of Technology, Sydney, Australia.

[deRaedt-Bruynooghe88] De Raedt L. and Bruynooghe M. (1988). On Interactive concept-learning and assimilation. In D. Sleeman (Ed.), *Proceeings of the Third European Working Session on Learning.* (pp. 167-176). Pitman.

[Esposito93] Esposito F., Malerba D. and Semeraro G. (1992). Classification in Noisy Environments Using a Distance Measure Between Structural Symbolic Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14, 3.

[Gams89] New Measurements Highlight the Importance of Redundant Knowledge. In *European Working Session on Learning (4th : 1989 : Montpeiller, France).* Pitman.

[Kononenko92] 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.

[Kruskal78] Kruskal W.H. and Tanur J.M. (1978). *International encyclopedia of statistics.* New York, NY: Free Press.

[Kwok90] Kwok S. and Carter C. (1990). Multiple decision trees. *Uncertainty in Artificial Intelligence, 4*, 327-335.

[Muggleton89] 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-Feng90] Muggleton S. and Feng C. (1990). Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory.* Tokyo. Ohmsha Press.

[Pazzani-Brunk91] 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-Kibler91] Pazzani M. and Kibler D. (1991). The utility of knowledge in inductive learning. *Machine Learning, 9, 1*, 57-94.

[Quinlan90] Quinlan R. (1990). Learning logical definitions from relations. *Machine Learning, 5, 3.*

[1] segal-etzioni94 Segal R. and Etzioni O. (1994). "Learning Decision Lists Using Homogoneous Rules" in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA: AAAI Press.

[Smyth92] 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.