A tale of two representations*

Sandip Sen
Department of Mathematical & Computer Sciences
University of Tulsa
600 South College Avenue, Tulsa, OK 74104-3189, USA
e-mail: sandip@kolkata.mcs.utulsa.edu

Abstract

Attribute based concept learning has been an active area of machine learning research over the past decade. A number of proposed concept learners are rule-based systems, whose efficiency depends largely on the representation of the rules. We analyze two attribute-value rule representations for the compactness of concept definition. A spectrum of concepts, with different numbers of conjuncts and disjuncts in their definition, is used to evaluate the effectiveness of the chosen representations. Our analysis shows that neither of the representations is dominant over the other, and it is beneficial to switch representations with varying concept complexity. Experimental verification of the predicted tradeoff is provided. The relatively novel of the two representations is found to be better, in general, at concisely representing concepts than the more commonly used representation. We present a formula to choose between these representations, which uses an estimate of the complexity of the concept to be learnt.

1 Introduction

Supervised concept learning from preclassified examples involves taking a set of input instances and trying to inductively infer the target concept. Approaches to solving classification problems are varied, including symbolic [Qui86], neural [RHW86], evolutionary [Hol86], etc. A large number of automated classification approaches use pattern matching rules to represent the target concept. In these systems, the accuracy and speed of learning depends largely on the appropriateness of the representation chosen to describe rules and instances. Since classification problems abound in real world practical applications including business, industry, scientific, and engineering domains [MA92], analysis and development of representation schemes that allow for more accurate rule learning with relatively few training examples will have a significant commercial impact.

In this paper we will address one particular aspect of the effect of representation on concept learning, namely that of the minimal size of the concept description possible given a concept and a representation. We assume that the more compact the description of a concept, the easier is the task of learning the concept. We recognize that several other factors do affect the accuracy and speed of learning the concept, but we believe that other things being equal, it is preferable to use representations that yield more compact concept descriptions. This paper is an investigation into two particular representations (Section 3), and their prospective usefulness for inductive concept acquisition. We analyze a number of concepts of varying complexity (with respect to their DNF representation) and calculate the size of the corresponding concept descriptions in the two representations of our choice (Section 4). To confirm our notion that the size of concept description plays an important role in determining the difficulty of learning the concept, we perform experiments to learn these concepts with a simple classifier system (described in Section 5); the results substantiate our intuitions (Section 6). We also present some analysis to help choose one of the two representations for concept learning if one has a rough estimate of the complexity of the target concept (Section 7). We summarize our findings at the end of the paper (Section 8). Our research is complimentary to the work of Booker [Boo91], Schuurmans and Schaeffer [SS89] and others, who are more concerned about the representational adequacy of the commonly used classifier language. Though experiments reported in this work have used a classifier system for learning, we believe that the results evaluating the usefulness of the two chosen representations for concept description are applicable in general to other rule-based systems.

2 Application areas

Classification problems are ubiquitous in nature, and are encountered in almost every practical and theoretical discipline. We would like to stress the general

*With due apology to Charles Dickens.

applicability of the classification paradigm, and hence will not focus our project to any particular discipline. However, to corroborate our claim that our system can be applied to a wide variety of real-life application domains, we list below a number of interesting classification problems gleaned from the University of California, Irvine repository of machine learning databases:

Medical applications: The following is a list of some medical applications of classification:

- classifying breast-cancer patients
- classifying echocardiogram traces
- classifying hepatitis patients
- classifying patients with Adult Respiratory Distress syndrome (ARDS)

Business applications: The following lists some applications of classification in business:

- screening credit card applications
- predicting housing prices in Boston suburbs
- predicting final settlements in labor negotiations in Canadian industry
- analyzing student loans

Scientific applications: In the following we list some scientific classification problems:

- classifying of radar returns from the ionosphere
- identifying promoter and splice-junction gene sequences
- classify solar flares that occur in a 24 hour period
- classifying low resolution spectrometer data

Industrial and Engineering applications: The following are some of the possible applications of classification techniques in the industrial and engineering disciplines:

- predicting automobile fuel consumption rates
- analyzing design of Pittsburgh bridges
- classifying computer hardware performance in terms of cycle time, memory size, etc.
- identifying types of glass from their oxide contents
- fault diagnosis of electromechanical devices
- classifying mushrooms as poisonous or edible in terms of their physical characteristics
- analysis of launch temperature Vs. O-ring stress on USA Space shuttle Challenger
- categorization of Landsat Satellite images

Our work on identifying rule representations that allow the acquisition of more accurate classification knowledge with less number of training instances will help develop more effective applications for the above-mentioned problem domains.

3 Two representations

We now introduce the two particular attribute-value representations that we propose to evaluate in the rest of the paper. We first introduce a representation for instances of the problem domain. An instance is represented as an ordered set of pairs, where the number of such pairs is equal to the number of attributes in the domain, and each pair consists of an attribute and one of the possible values that attribute can assume. For example, an instance in a domain containing 4 attributes, each of which can take on any of the 4 values 1,2,3,4 can be represented as $\{(x_1, 2)(x_2, 4)(x_3, 3)(x_4, 1)\}$, where x_i denotes the ith attribute.

3.1 Value representation

This particular representation is perhaps the most common representation used by rule-based concept learning systems. Each rule consists of a condition and a classification component, separated by a ":" symbol. The prediction part contains one of the several different defined classifications in the domain. The condition part is a conjunction of several terms, each term corresponding to one of the attributes in the domain. Each term is of the form $x = c$ where x is an attribute and c is one of the possible values that x can assume, or it is a special symbol called "don't care". A rule in this format will match an instance of the problem domain if corresponding attributes in the instance and the rule is bound to the same value, or if the latter is bound to the don't care symbol. For example, the instance given at the start of Section 3 will be matched by the rule

$$(x_1 = 2) \wedge (x_2 = \#) \wedge (x_3 = \#) \wedge (x_4 = 1) : 1,$$

where $\#$ is the don't care symbol, but will not be matched by the rule

$$(x_1 = 2) \wedge (x_2 = \#) \wedge (x_3 = 2) \wedge (x_4 = 1) : 1.$$

The use of the don't care symbol allows the formation of generalized rules that can correctly classify a number of problem instances. Effectively, this is a means for forming equivalence classes over the input space. This representation is good for representing concepts in disjunctive normal form, where each of the clauses involve a conjunction of a number of terms, each term containing an equality test on an attribute different from other attributes in the clause. Each clause is represented by a different rule, and a set of rule together defines the concept. It is difficult to represent concepts containing terms like $x_1 = x_2$ (or any term that relates two different attributes), or for representing terms with operators other than the standard equality operator (like $>, \geq, <, \leq, \neq$).

3.2 Bit representation

This representation is not as intuitive as the one above but, as we will show, can represent some concepts succinctly. We first noticed it in the work of De Jong and Spears [DS91], where they called it a *fixed-length* representation. We have chosen to adopt another name because the rules in the other representation being discussed are also fixed length rules. As above, each rule is represented by a condition and a classification part, separated by the ":" symbol. The prediction part contains one of the several different defined classifications in the domain. The condition part is a conjunction of several terms, each term corresponding to one of the attributes in the domain. Each term is of the form $x = b_1 \ldots b_m$, where m is the cardinality of the ordered domain over which the attribute x is defined, and $b_i =$ 1 or 0 suggests whether the rule can or cannot match a problem instance in which the attribute x is bound to the ith value in its ordered domain. A rule matches an instance if for every attribute in the instance, the attribute is bound to a value in its domain such that the corresponding bit is 1 in the bit-string associated with this attribute in the rule. As an example, the instance given at the beginning of section 3 is matched by the rule

$$(x_1 = 0100) \wedge (x_2 = 1111) \wedge (x_3 = 1111) \wedge (x_4 = 1000) : 1,$$

but not by the rule

$$(x_1 = 0100) \wedge (x_2 = 1111) \wedge (x_3 = 0100) \wedge (x_4 = 1000) : 1,$$

where each attribute is defined over the domain consisting of the 4 ordered elements 1,2,3,4.

Each rule given above is the same as the rules given in the previous subsection, though represented using the bit representation. Don't care symbols are implicitly encoded in this representation by setting all the bits associated with the relevant attribute. This representation is good for representing concepts in conjunctive normal form (CNF), where each of the clauses involve a disjunction over a number of terms, each term containing one or more equality tests on the same attribute (internal disjunctions), but for different values in its domain (this type of multiple equality testing can represent operators like $>, \geq, <, \leq, \neq$).

This representation can also be used to represent DNFs (as one can easily represent each rule in the value representation by a corresponding rule in the bit representation); the issues to be concerned about in those cases are the same as those cited for the value representation. In general, one rule in bit representation will translate into a number of rules in value representation. The bit representation has problems with representing relations between attributes. Another note of caution is in order: the bit representation described here is very different from the bit encoding of the attribute values that is often used in conjunction with standard genetic algorithms.

4 Evaluating tradeoffs

We now introduce quantitative metrics to evaluate concept definitions obtained by the value and bit representations. We also identify a class of problems of varying complexity to be used to illustrate the tradeoffs in using these representations for concept learning.

4.1 Information Units

We are interested in the compactness of concept representation. In order to evaluate alternative representation techniques by this criterion we define the *Information Units* (**IU**) criterion as the number of symbols contained in all the rules required to represent a concept using any particular representation (we do not however count the attribute names in the rules as separate symbols). More formally, given a set of rules $\mathcal{R}$ used to represent a concept defined over N attributes, where the ith attribute is defined over a domain of cardinality s_i, the number of information units for the two representations discussed above are defined as follows:

1. for the value representation, $\mathbf{IU}(\mathcal{R}) = |\mathcal{R}| * (N + 1)$,
2. for the bit representation, $\mathbf{IU}(\mathcal{R}) = |\mathcal{R}| * (\sum_{i=1}^{N} s_i + 1)$.

For example, if we consider any one rule given in the previous section, the number of **IU**s contained in a rule using the value representation is 5, whereas the number of **IU**s contained in a rule using the bit representation is 17.

4.2 Problem Set

Now we introduce the set of problems we will use to evaluate the alternative representations already discussed. The idea of this problem set is largely borrowed from the work of De Jong and Spears [DS91], but the exact concepts may or may not be different from theirs (they do not specify the exact concepts used). We chose a 4-attribute classification domain with each problem instance labeled as one of two possible classes (0 or 1). Each attribute was defined over an ordered domain $\{1, 2, 3, 4\}$. We have considered 12 concept learning problems of varying complexity. The concepts are represented in their disjunctive normal form (DNF). Complexity is measured by the number of disjunctive clauses and the number of conjunctive terms per disjunctive clause in the concept description. Each of the 12 concepts is labeled as nDmC, where n is the number of disjuncts (varied from 1 to 4) and m is the number of conjuncts (varied from 1 to 3). Note that these numbers represent disjuncts and conjuncts to correctly classify positive instances (ones with classification 1) only. A number of other rules will be required to classify the negative instances (ones with classification 0) correctly. For any value of n and m we chose one concept from a number of concepts available of the complexity specified by the choice of n and m.

Given a concept, we calculated by hand the minimal set of rules to properly classify the negative and positive instances using both the representations. There was one restriction imposed on the construction of rule sets: we decided that rules matching positive and negative instances should be mutually exclusive (that is, for a given problem instance we never have two matching rules that predict different classifications). This choice was made because we did not want to consider the effects of the learning and conflict resolution that is required to handle inconsistent rule bases. We understand that our choice eliminated possibilities of building default hierarchy of rules, etc., but we believe those issues are only partially relevant (particularly as the number of possible classes gets larger) to the issue of compactness of concept description as determined by the representation itself. We also require that the minimal rule set do classify both positive and negative instances correctly (instead of matching only positive instances and classifying examples as negative instances if it cannot be matched), as such a method is ineffective for problems with more than two classes.

Given the minimal set of rules obtained with the two representations for the same concept, the number of information units in these rule sets are compared to evaluate the effectiveness of the two representations for concisely defining the concept. In Table 1 we present two concepts corresponding to 1D1C and 2D1C, the minimal rule sets for the value and bit representations and the corresponding information units contained in them. For the sake of simplicity, in this and following tables, we have omitted the attribute names from the rule descriptions and separated the bindings for different attributes by commas (they also represent the conjunctions between the neighboring attributes in the original rules). As we increase the number of disjuncts keeping the number of conjuncts fixed at 1, the increase in **IU**s for the minimal rule set obtained with the value representation is seen to be much more than that obtained with the bit representation. This is because it is easy to represent the inequality operator using the bit representation (see the third rule for the 2D1C concept) whereas in the value representation we have to have separate rules corresponding to each of the other possible values. Whenever there are multiple disjunctions for any attribute in the concept description (like $(x_1 = 2) \vee (x_1 = 3) \vee (x_1 = 4)$) what can be implemented using one rule in bit representation will take several rules in value representations (the number of **IU**s will vary depending on the number of attributes and the size of their domains). This problem for value representation is made more acute as the number of attributes with multiple disjunctions in the problem description increases, because we have to represent all possible combinations of the different disjunctions of these attributes (see the minimal set of rules for the second concept using value representation).

In table 2 we present similar data as above over a range of concepts of differing complexity. For each of them we present the **IU**s for the minimal rule set formed by each representation. Additionally, we only present the minimal set of rules formed using the bit representation; we leave out the minimal rule set for the value representation in view of limited space, because in several cases the number of rules is high. From the data given in the table it is obvious that neither of the representations is better than the other in concisely representing concepts across the range.

On the whole, it seems that as the number of conjuncts per clause (in the DNF concept description) increases, the minimal rule set size obtained using both the representations increases. A large part of the increase in the case of the value representation is to cover negative instances of the concept. The increase in the number of minimal rule set size for bit representation is much smaller (as the inequality operator can be represented more succinctly), but the overall increase in the number of **IU**s is a little higher because of the longer length of each rule in this format.

As the number of disjunctive clauses (in the DNF concept description) grows, the minimal rule set size and the corresponding **IU** values for the value representation increases exponentially. This increase can again be attributed to the inability of the value representation to represent the inequality operator. The problem is accentuated in this case because as the number of attributes requiring the inequality operator increases, all possible combinations of different values for each of these attributes have to be represented by

Concept: If $(x_1 = 1)$ then 1, otherwise 0; complexity 1D1C			
Value representation		Bit Representation	
Minimal rule set	**IU**	Minimal rule set	**IU**
1, #, #, # : 1 2, #, #, # : 0 3, #, #, # : 0 4, #, #, # : 0	20	1000, 1111, 1111,1111 : 1 0111, 1111, 1111,1111 : 0	34

Concept: If $(x_1 = 1) \vee (x_2 = 1)$ then 1, otherwise 0; complexity 2D1C			
Value representation		Bit Representation	
Minimal rule set	**IU**	Minimal rule set	**IU**
1, #, #, # : 1 #, 1, #, # : 1 2, 2, #, # : 0 2, 3, #, # : 0 2, 4, #, # : 0 3, 2, #, # : 0 3, 3, #, # : 0 3, 4, #, # : 0 4, 2, #, # : 0 4, 3, #, # : 0 4, 4, #, # : 0	55	1000, 1111, 1111,1111 : 1 1111, 1000, 1111,1111 : 1 0111, 0111, 1111,1111 : 0	51

Table 1: Representation of two simple concepts using value and bit representations.

separate rules for accurate classification. Let us illustrate this by referring to table 1. For the first concept, the second rule in the bit representation is equivalent to "if $(x_1 \neq 1)$ then 0"; the same information is represented by 3 distinct rules in the value representation (the last three rules), one for each of the other values that x_1 can assume. For the second concept described in this table, the third rule in the bit representation is equivalent to "if $(x_1 \neq 1) \wedge (x_2 \neq 1)$ then 0"; to represent the same information using the value representation we require 9 different rules (the last 9 in the rule set) corresponding to every combination of the other values (except 1) that x_1 and x_2 can assume. One immediate remedy that can be suggested is using both equality and inequality operators within the value representation. That leads to an increase in the search space and also the number of **IU**s to be learnt. Even with this additional operator, the value representation lacks the capability of the bit representation that lets the latter represent arbitrary subsets (for example, $(x_1 = 1) \vee (x_1 = 4)$ can be represented in the bit representation as $(x_1 = 1001)$).

From the above, we deduce that if the DNF of the target concept contains only 1 conjunctive term per clause, the value representation will produce more compact representation. But as the number of clauses increase the bit representation produces more and more compact representations of the concept description compared to the value representation.

The compactness with which concepts can be described is just one aspect of evaluating a representation. The actual effectiveness of learning a concept is also determined to a considerable extent by the learning algorithm used (and the associated bias), the experimental method used (batch versus incremental learning), the metric of evaluation (how quickly the concept is learned, how accurate is the learned description), and the size of the concept space as defined by the representation. In order to assess the generality of our analysis we decided to run experiments with NEWBOOLE, a simple classifier system used for supervised, incremental concept learning. The primary metric we were interested in is the accuracy of the learned description. We used speed of learning as a secondary metric to choose between the representations, when they fared equally well on the primary metric. In the following we present a brief description of NEWBOOLE, and then present our experimental results.

5 NEWBOOLE

NEWBOOLE [BPSW90] is a simple stimulus-response classifier system in which receiving an input from the environment and reacting to it takes place in a single time step. It is an improvement over the BOOLE [Wil87] system, and has been used successfully in a number of supervised classification problem domains [BP91, Sen93]. NEWBOOLE uses a population of classifiers (equivalent to bit-level zero order rules) which are evolved using genetic operators modeled after Darwinian principles. The rule representations used in this paper are as described previously. Associated with each classifier is a numeric strength attribute, which is a measure of the usefulness of the classifier. At the start of a run, all the classifiers are initialized randomly with approximately δ% of the attribute values set to # (value representation) or 1 (bit representation), and their strengths are set to a default value, s_I. In each trial, an input chosen randomly from the input space is presented to the system, the system responds with an output which is then matched with the desired output and an appropriate feedback is given to the system based on its performance in this trial. The system adjusts itself based on the feedback and then genetic operators are used to evolve better rules from present ones. The different components of the system is described in some detail below.

On receiving an input, the **performance component** generates a matchset ($\mathcal{M}$) of classifiers whose taxon matches the input. Each matched classifier has a record of the number of times it predicted correctly and the number of times it predicted wrongly when it matched an input string in the past. Three evaluation criteria is used to choose the classifier from $\mathcal{M}$ to provide the system answer: (1) preference is given to classifiers which predicted wrongly least often on matching an input, (2) for classifiers performing identically on the above criteria preference are given to classifiers who have predicted correctly most often when matching the input than others, (3) for classifiers performing identically on the above two criteria, preference is given to classifiers with higher strength. If there is still a tie, a classifier is picked randomly from those tied. The action of the chosen classifier is provided as the system response in that trial.

The **reinforcement component** is responsible for altering classifier strengths based on their performance in any given trial. It consists of the following steps: 1) Form the set, $\mathcal{C}$, a subset of $\mathcal{M}$, whose action is the same as the desired output. $\mathcal{C}$ and Not $\mathcal{C} = \mathcal{M} - \mathcal{C}$, represent the set of correct and incorrect classifiers respectively. 2) Deduct a fraction e (bidding tax) from the strengths of each member of $\mathcal{C}$ and distribute a payoff quantity R among all the members of $\mathcal{C}$ in a manner outlined later. 3) Deduct a penalty fraction, p, from the strengths of each member of Not $\mathcal{C}$.

A biased distribution, D, is used to distribute the payoff R to the members of $\mathcal{C}$. The distribution can be changed by choosing a generality bias parameter, G, to favor more general or more specific rules depending on the problem. We define the generality of a classifier as the fraction of its genes (string position values) which are don't cares, and is therefore a measure of the number of distinct input strings that the classifier is able to match. So, generality of the ith classifier is given by

$$g_i = \frac{\text{number of \#'s in } i}{\text{taxon length}}.$$

The portion of the payoff R that goes to classifier $i \in \mathcal{C}$ is given by

$$R_i = \begin{cases} \frac{d_i}{\sum_{j \in C} d_j} \times R & \text{if } w_i = 0 \\ 0 & \text{otherwise} \end{cases}$$

where, $d_i = 1 + G \times g_i$, and w_i is the number of times the i^{th} classifier has predicted wrongly when matching an input.

The **discovery component** uses a genetic algorithm [Gol89] to generate more useful classifiers over time. The parameter ρ represents the number of offsprings produced per generation using the genetic algorithm. For each new offspring to be produced the following steps are carried out: 1) A parent P_1 is selected using a probability distribution over strengths of all classifiers.parent P_2 is chosen in the above manner. Two crossover points are selected randomly by treating the classifier strings as rings, portions of the strings in between are exchanged, and one of the two new structures is randomly chosen as the offspring. 3) If P_2 was not chosen then the new offspring is produced from a copy of P_1. 4) With probability $pmut$, mutate each position of the offspring. Mutation flips the value of a string position equiprobably to one of the other possible values. 5) Set the strength of the newly created offspring to a constant s_I. 6) Choose randomly from the classifiers which have predicted wrongly the maximum number of times, and replace it with the new offspring. Additionally, if no classifier matches an input, a new classifier is formed by generalizing from that input, and is inserted into the population by replacing an existing classifier.

6 Experiments

For each of the target concepts described in tables 1 and 2 we ran experiments with NEWBOOLE using both value and bit representations. The system parameters were set as follows: population size = 400, crossover probability = 0.9, mutation probability = 0.001, individuals replaced per trial = 4, initial strength of classifiers = 100, payoff for correct answer = 1000, penalty fraction = 0.95, bidding tax = 0.1, generality bias coefficient = 0.1, initial don't care percentage = 50. For each concept there are 256 instances. At each time step (or trial) one instance is drawn at random from this set and presented to the system for classification. At intervals of 10 trials we calculate both the incremental and the batch accuracy of the system in classifying problem instances. Incremental accuracy is measured as the percentage of correct classification over the last 10 trials. Batch accuracy is measured by turning off the learning component of the system, and measuring its classification accuracy over all 256 instances.

We now present the results obtained from some of these experiments. In figure 1 we present the learning curves obtained by plotting classification accuracy rates over the length of the experiment (500 trials). Each data point in the plots is averaged over 10 runs of the system with different random seeds to form the initial rule set. We present data from experiments with concepts of complexity 1D1C, 4D1C, and 1D2C. The results match the trends in the predicted tradeoffs in **IU**s for the two representations (as presented in tables 1 and 2). For the concept belonging to 1D1C ("if $(x_1 = 1)$ then 1, otherwise 0") the value representation produces faster and more accurate learning than the bit representation. On the other extreme, for the concept belonging to 4D1C ("if $(x_1 = 1) \vee (x_2 = 1) \vee (x_3 = 1) \vee (x_4 = 1)$ then 1, otherwise 0"), the bit representation produces faster and more accurate learning than the value representation. In these cases, the ability of NEWBOOLE to learn the concepts in an incremental setting is found to be related closely to the compactness of the concept description possible with the representation used. However, the gap between the learning curves is found to be much narrower for the concept belonging to 1D2C ("if $((x_1 = 1) \vee (x_2 = 1))$ then 1, otherwise 0"). This is because positive examples are much rarer than negative examples and both the systems learn very general rules (predicting negative classification) to cover the entire space. This unfortunate trend is noticed in other scenarios where there is a major imbalance between the number of positive and negative instances of the concept. In these scenarios, NEWBOOLE classifies almost all the problem instances to be of the majority class and still performs extremely well (at about 99% accuracy). Its accuracy over the whole set is found to be quite good (as there are few instances of the minority class), but the system really has not found the correct concept description. We believe that other concept learners like ID3, which memorize every single instance if necessary, will produce learning curves more consistent with our calculations as presented in tables 1 and 2.

7 Further analysis

In this section, we present a general analysis of concept classes for which a more compact description is obtained with the bit or the value representation. We assume a minimal set of rules for a concept in the bit representation, and generate from it a minimal set of rules in the value representation. We then count the number of **IU**s contained in each of these rule sets, and use these counts to choose between the two representations for this particular concept. We find that a rough estimate of the complexity of an imprecisely understood concept can be used to choose a more appropriate representation to learn the concept.

We consider concepts defined over N attributes, where each attribute can take on any of M different values. The number of possible instances is therefore M^N, each of which is classified into one of the two classes, 0 or 1. Let the minimal bit representation of any such concept require K rules to define the concept. Let there be on an average a attributes per rule for which the number of 1s in the bit string equated to them is greater than 1 but less than M (in the following analysis we assume $a > 0$). Let there be on an average d 1s per rule which are neither part of a don't care, nor the only 1 in a bit string equated to an attribute. Consider the rule "$((x_1 = 0010) \wedge (x_2 = 1001) \wedge (x_3 = 1111) \wedge (x_4 = 1110)) : 0$"; here, $a = 2$ (corresponding to x_2 and x_4) and $d = 5$ (adding the number of 1s in the bit strings equated to x_2 and x_4). In order to estimate the number of rules required in the value representation to represent this concept, we further assume that these d 1s are evenly distributed among the a non-don't care (don't care attributes have M bits equal to 1 in the corresponding bit string) and non-specific (specific attributes have 1 bit equal to 1 in the corresponding bit string) attributes (NDNS attributes). With these assumptions, we see that the minimal number of rules required to define this concept using the value representation is $K * \left(\frac{d}{a}\right)^a$. The corresponding number of **IU**s is $K * N * \left(\frac{d}{a}\right)^a$. The number of **IU**s contained in the minimal rule set in the bit representation we started with is $K * N * M$. Therefore, we should use the value representation or the bit representation to learn the concept depending on whether $\left(\frac{d}{a}\right)^a$ is significantly lower than M or significantly greater M respectively. When these two terms are almost equal, any one of the representations can be used. Let us consider the expression $\left(\frac{d}{a}\right)^a$ in more detail. Given that $d > a$, this term increases as a and d increases. The maximum value for this term is $\left(\frac{N*(M-1)}{N}\right)^N$, when all the attributes have exactly one 0 in the bit strings equated to them. Consider for example the rule "$((x_1 = 1110) \wedge (x_2 = 1101) \wedge (x_3 = 0111) \wedge (x_4 = 1110)) : 1$" in bit representation; this rule translates into $\left(\frac{4*(4-1)}{4}\right)^4 = 81$ different rules in the value representation.

To summarize, if the minimal rule set definition of a concept using the bit representation is expected to contain a large number of NDNS attributes, each of which contains a large number of 1s (alternative values that these attributes can adopt), the bit representation of the concept will be much more compact than the value representation. One can similarly identify situations when value representation produces a more compact concept description, or when the two representations are roughly equivalent in the size of the concept representation.

8 Conclusions

In this paper we have analyzed two different representations, value and bit, for the compactness of concept descriptions they can produce. Initial analysis suggested that for concepts in DNF, value representation produced more compact definitions if the DNF had a very small number of clauses, but the bit representation would produce smaller descriptions as this number increases beyond a threshold. Experiments conducted with a classifier based incremental concept learner showed that learning and accuracy rates varied proportionately with concept description size if both positive and negative instances were present in sufficient numbers. We also provide an analysis to choose between these two representations for learning a concept, given an approximate estimation of the nature of the concept. Surprisingly, for a large variety of concepts, the not so common bit representation seems to be a better choice for cocept learning, as compared to the popular value representation. This finding will enable the choice of a more appropriate representation for machine learning systems in real-world classification problems, thus providing considerable savings in revenues on existing applications as well as making it possible to develop new applications.

References

[Boo91] Lashon B. Booker. Representing attribute-based concepts in a classifier system. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 115–127, 1991.

[BP91] Pierre Bonelli and Alexander Parodi. An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 288–295, San Mateo, CA, 1991. Morgan Kaufman.

[BPSW90] Pierre Bonelli, Alexander Parodi, Sandip Sen, and Stewart W. Wilson. Newboole: A fast gbml system. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 153–159, San Mateo, CA, 1990. Morgan Kaufman.

[DS91] Kenneth A. DeJong and William M. Spears. Learning concept classification rules using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656, 1991.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Hol86] John H. Holland. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R.S. Michalski, J.G. Carbonell, and T. M. Mitchell, editors, *Machine Learning, an artificial intelligence approach: Volume II.* Morgan Kaufman, Los Alamos, CA, 1986.

[MA92] P.M. Murphy and D.W. Aha. Uci repository of machine learning databases [machine-readable data repository], 1992.

[Qui86] Ross J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA, 1986.

[Sen93] Sandip Sen. Improving classification accuracy through performance history. In *Proc. 5th Intl Conf on Genetic Algorithms*, 1993.

[SS89] D. Schuurmans and J. Schaeffer. Representational difficulties with classifier systems. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 328–333, San Mateo, CA, 1989. Morgan Kaufman.

[Wil87] Stewart W. Wilson. Classifier systems and the animat problem. *Machine Learning*, 2:199–228, 1987.

Concept: If $(x_1 = 1) \vee (x_2 = 1) \vee (x_3 = 1)$ then 1, otherwise 0; complexity 3D1C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
150	1000, 1111, 1111,1111 : 1 1111, 1000, 1111, 1111 : 1 1111, 1111, 1000, 1111 : 1 0111, 0111, 0111, 1111 : 0	68

Concept: If $(x_1 = 1) \vee (x_2 = 1) \vee (x_3 = 1) \vee (x_4 = 1)$ then 1, otherwise 0; complexity 4D1C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
475	1000, 1111, 1111,1111 : 1 1111, 1000, 1111, 1111 : 1 1111, 1111, 1000, 1111 : 1 1111, 1111, 1111, 1000 : 1 0111, 0111, 0111, 0111 : 0	85

Concept: If $(x_1 = 1) \wedge (x_2 = 1)$ then 1, otherwise 0; complexity 1D2C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
35	1000, 1000, 1111,1111 : 1 1111, 0111, 1111, 1111 : 0 0111, 1111, 1111, 1111 : 0	51

Concept: If $(x_1 = 1) \wedge (x_2 = 1) \wedge (x_3 = 1)$ then 1, otherwise 0; complexity 1D3C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
50	1000, 1000, 1000,1111 : 1 0111, 1111, 1111, 1111 : 0 1111, 0111, 1111, 1111 : 0 1111, 1111, 0111, 1111 : 0	68

Concept: If $((x_1 = 1) \wedge (x_2 = 1)) \vee ((x_1 = 2) \wedge (x_2 = 2))$ then 1, otherwise 0; complexity 2D2C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
80	1000, 1000, 1111,1111 : 1 0100, 0100, 1111, 1111 : 1 1011, 0111, 1111, 1111 : 0 0111, 1011, 1111, 1111 : 0	68

Concept: If $((x_1 = 1) \wedge (x_2 = 1) \wedge (x_3 = 1)) \vee ((x_1 = 2) \wedge (x_2 = 2) \wedge (x_3 = 2))$ then 1, otherwise 0; complexity 2D3C		
Value representation (**IU**)	Bit Representation	
	Minimal rule set	**IU**
320	1000, 1000, 1000,1111 : 1 0100, 0100, 0100, 1111 : 1 0111, 1011, 1111, 1111 : 0 1011, 0111, 1111, 1111 : 0 1011, 1011, 0111, 1111 : 0 0111, 0111, 1011, 1111 : 0	102

Table 2: Concepts of varying complexity and their bit representation.

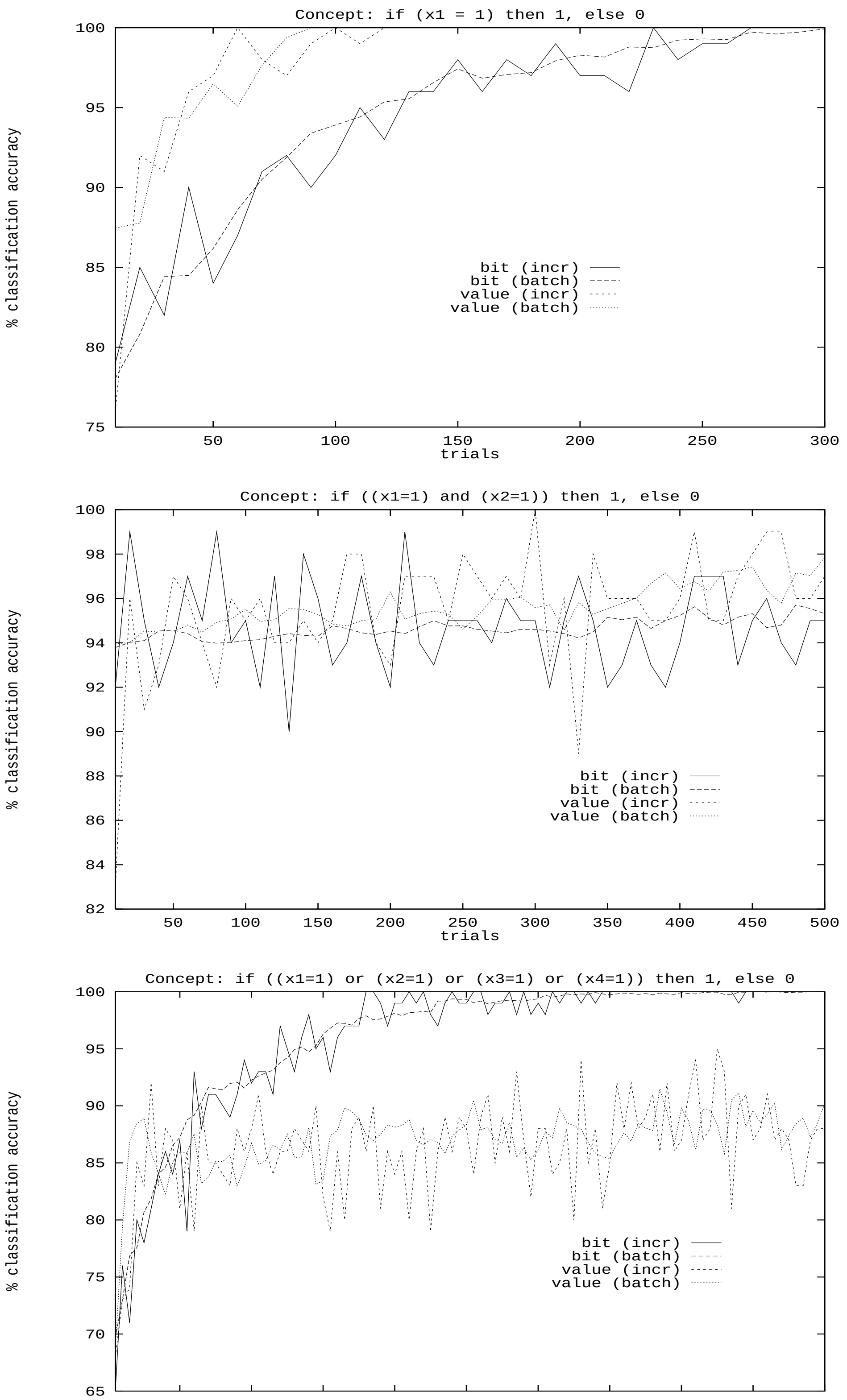

Figure 1: Learning curves for some concepts with value and bit representations.