Improved Comprehensibility and Reliability of Explanations via Restricted Halfspace Discretization

Klaus Truemper

Department of Computer Science,
University of Texas at Dallas,
Richardson, TX 75083,
U.S.A.

Abstract. A number of two-class classification methods first discretize each attribute of two given training sets and then construct a propositional DNF formula that evaluates to *True* for one of the two discretized training sets and to *False* for the other one. The formula is not just a classification tool but constitutes a useful explanation for the differences between the two underlying populations if it can be comprehended by humans and is reliable. This paper shows that comprehensibility as well as reliability of the formulas can sometimes be improved using a discretization scheme where linear combinations of a small number of attributes are discretized.

Key words: Discretization, Logic, Explanation, Comprehensibility, Reliability

1 Introduction

Suppose we are to explain the differences between two populations $\mathcal{A}$ and $\mathcal{B}$. In the well-known approach assumed here, we take two training sets A and B from the populations $\mathcal{A}$ and $\mathcal{B}$, discretize the attributes of the training sets, and construct a propositional disjunctive normal form (DNF) formula that evaluates to *True* for one of the two discretized training sets and to *False* for the other one. Example methods carrying out these steps in various ways are described in [1, 2, 4, 8, 10, 11, 17–19, 24, 25, 47]. The formula can be viewed as a useful explanation of the differences between the populations if (1) the formula can be comprehended and interpreted by humans, and (2) it reliably predicts membership in the two populations. This paper shows that a certain discretization where linear combinations of a small number of attributes are discretized, may help in the construction of comprehensible and reliable explanations.

We first discuss the size and comprehensibility of formulas.

2 Size and Comprehensibility of Formulas

Human comprehension of data or statements is an extensively covered topic of Neurology and Psychology. One of the key concepts is *chunk*, defined in [20] as a collection of concepts that are closely related and that have much weaker connections with other concurrently used concepts. The seminal paper [40] defines a "magical number seven, plus or minus two" of chunks as limit of short-term memory storage capacity. Subsequent work refined the main claim of [40]. For a detailed review, see [20], which argues for a "magical number 4" of chunks. In related research, [31] establishes a limit of 4 for the number of variables humans can process. In [32] an integrated treatment of *working memory capacity* and *relational capacity* is proposed that generalizes the above results. The reference concludes that working memory is limited to approximately 3-4 chunks, and that the number of variables involved in reasoning is limited to 4. We rely on these bounds in our quest for comprehensible explanations.

2.1 Formula Size

Let us define the *formula size FS* of the logic formulas of interest here. The formulas are always in *disjunctive normal form* (DNF). Such a formula is a disjunction of DNF *clauses*, which in turn are conjunctions of literals. In this paper, each literal is a linear inequality of the form $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$, where z is a vector of attribute variables, b is a vector of constants, and α is a scalar. An example of a DNF formula is $[(x < 5) \wedge (y > 7)] \vee (z > 1) \vee [(y < 1) \wedge (z < 0)]$, with DNF clauses $[(x < 5) \wedge (y > 7)]$, $(z > 1)$, and $[(y < 1) \wedge (z < 0)]$.

Consider a literal $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$ with $k \leq 4$ nonzero entries b_j in the vector b. In agreement with the conclusions of [32], we assume that humans can readily understand the interaction of the items represented by the k terms $b_j \cdot x_j$ with nonzero b_j and convert that information into one chunk that allows further reasoning. For the case $k \leq 3$, graphical representation of the corresponding halfspace in k-dimensional Euclidean space can assist in that task. We call the translation of $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$ with $k \leq 4$ nonzero b_j into just one chunk *elementary chunking*. Any other chunking is considered *non-elementary*.

The formula size FS of a DNF formula where each literal is of the form $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$ with $k \leq 4$ nonzero b_j is defined to be equal to the number of literals in the formula. Thus, it is equal to the number of chunks that can be derived by elementary chunking. We handle Boolean variables w in the above format by encoding the literals w and $\neg w$ by the literals $(x > 0.5)$ and $(x < 0.5)$, respectively, using a $\{0, 1\}$ variable x. Each such term contributes 1 to the FS value of the formula. There are other ways to define formula size, using different notions of chunking. We have chosen the above definition mainly because elementary chunking is consistent with [32].

In the next section, we encounter example logic formulas that only contain literals of Boolean variables. As a matter of brevity and convenience, we skip translation to the format where literals are inequalities and use the formula directly. FS of such a formula is defined to be equal to the number of literals of the formula, which would be the FS value of the appropriately translated formula version.

2.2 Comprehension Difficulty

The approach of this section is motivated by the experiments reported in [31]. We want a measure of difficulty of comprehension of DNF formulas when such formulas serve as explanations. We must first agree on what comprehension means. For our purposes, a reasonable definition is that the person who supposedly has comprehended a formula has indeed integrated the logic information implied by the formula into his/her relevant knowledge. For example, suppose a formula has the value *True* if a patient benefits from a certain medication, and has the value *False* otherwise. A physician has comprehended the formula if he/she has integrated the logic information into the treatment decision process.

We determine the *comprehension difficulty* CD of a formula S experimentally follows. Suppose n persons are proficient in a knowledge domain X. Let S be a DNF formula that applies to X. We give each person the formula S and ask whether S is compatible with X. Mathematically, answering the question is equivalent to deciding if $S \wedge X$ has a satisfying solution. Suppose that k persons answer the question correctly. If $k \leq n/2$, then human decision making essentially is not better than a random choice of answers. Accordingly, we estimate that comprehension of S by humans is not possible, and define CD to be ∞. Assume that $k > n/2$, and let t be the average time required to find an answer to the question. Reasonably, we want CD to be proportional to t and inversely proportional to the fraction $k/n - 0.5$, which measures the improvement of accuracy over random choice. We also want CD to have the value t if the n answers are 100% accurate. The following definition of CD meets these goals.

$$CD = \begin{cases} t/(2k/n - 1) & \text{if } k/n > 0.5 \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

2.3 Prediction of Comprehension Difficulty

We may be tempted to predict CD using FS. The main result of this section is that such a prediction generally is not possible. Two examples below prove that claim. But later we also show that, under certain conditions, a reasonable prediction of likely comprehensibility can be made.

In the first example, a formula S with large FS applies to a knowledge domain X such that CD is small. This result is not unexpected, since non-elementary chunking can have that effect. In the second example, a formula S with small FS applies to a knowledge domain X such that CD is large. This result can be expected for certain pathological situations. But the example shows that this situation can also occur in rather straightforward situations of S and X.

Example 1. The domain X covers grading of one exam. For $j = 1, 2, \dots,$ n, let the Boolean variable x_j have the value *True* if question j of the exam is answered correctly, and the value *False* otherwise. The formula S is to have the value *True* if the entire exam has at least two questions answered incorrectly. Regardless of the specific encoding of this condition, the size FS of S must grow at least linearly in n. But it is easy to check if $S \wedge X$ is satisfiable, since non-elementary chunking reduces that check to trivial counting. Thus, CD is small even when FS is large.

Example 2. In [31] it is shown that understanding food preference questions can be very difficult for humans. The following rephrasing of the results of [31] produces an example of low FS and high CD.

The questioning process of [31] concerns cakes in various forms and the related preferences by humans. We sketch the setting. There are four variables: *process*, *flavor*, *type*, and *richness*. Each variable can take on two values, as follows: *process* = *fresh* or *frozen*, *flavor* = *chocolate* or *carrot*, *type* = *iced* or *plain*, and *richness* = *rich* or *lowfat*. The knowledge domain X consists of bar graphs depicting the desirability of cakes with various features. For example, a value may be given for fresh cakes containing chocolate and covered with icing.

In each experiment, a person must evaluate a set of statements. The person is told that all statements of the set are correct except for the last one, which allows for two choices. The person must decide which of the two choices is the correct one, using the bar graphs of the knowledge domain X.

There are four types of sets of statements, called 2×2-*way*, 3-*way*, 2×3-*way*, and 4-*way*. The simplest case is of type 2×2-way. The most difficult one is of type 4-way. For the sake of brevity, we discuss only those two cases and omit the intermediate 3-way and 2×3-way cases.

Here is an example set of statements for the 2×2-way case. "People prefer *fresh* to *frozen* cakes. The difference [in the rating] depends on the flavor (*chocolate* vs *carrot*). The difference between *fresh* and *frozen* is (greater/smaller) for *chocolate* cakes than for *carrot* cakes." The person must decide which of the two options "greater" and "smaller" in the last statement is consistent with the knowledge domain X.

In each set of statements of the 4-way cases, differences of rating differences are mentioned and must be evaluated. An example set of statements is as follows. "People prefer *fresh* to *frozen* cakes. The difference [in the rating] depends on the flavor (*chocolate* vs *carrot*), the type (*iced* vs *plain*), and the richness (*rich* vs *lowfat*). The difference between *fresh* and *frozen* increases from *chocolate* cakes to *carrot* cakes. This increase is greater for *iced* cakes than for *plain* cakes. There is a (greater/smaller) change in the size of the increase for *rich* cakes than for *lowfat* cakes."

The number of cakes with various features listed in the bar graphs depends on the case. Specifically, for the 2×2-way case, each instance of the knowledge domain X lists 4 different cakes and their ratings. For the 4-way case, that number is 16.

Let us rephrase the setting and task using formulas that represent the statements in a naive way. For example, let $(\text{difference\_1}(\textit{fresh}, \textit{frozen}) > 0)$ be the single literal of a formula that evaluates to *True* if for all cakes with identical features except for *process*, the rating for *fresh* is higher than for *frozen*. This formula encodes the first statement of both examples for the 2×2-way and 4-way cases. In a similar fashion, each remaining statement can be encoded by just one literal that uses a new difference term. An exception are statements such as "The difference [in the rating] depends on the flavor (*chocolate* vs *carrot*)", which state a restriction that is implied by the bar graphs of the knowledge domain X and thus need not be translated. For the encoding of the last statement, one case of the two options is arbitrarily selected.

An entire set of statements can thus be represented by a DNF formula S that consists of just one DNF clause whose literals encode the statements. FS is equal to 2 for the 2×2-way case and equal to 4 for the 4-way case. Thus, FS is small in both cases. Consistency of S with the knowledge domain X holds if and only if the encoding of the last statement uses the correct option.

Reference [31] includes results for a number of experiments involving various situations. The average solution times are displayed in a graph. Visual inspection determines the average solution time for the 2×2-way case to be 25 sec and for the 4-way case to be 74 sec. Correctness counts of answers are provided for pairs of experiments. The average correctness rate implied by the counts is 1.00 for the 2×2-way case and 0.65 for the 4-way case. Using (1), the corresponding CD values are 25 and 247, respectively. The table below shows FS and CD for the two cases.

Formula Size and Comprehension Difficulty

Case	FS	CD
2×2-way	2	25
4-way	4	247

Evidently, the increase of FS from 2 to 4 causes an almost 10-fold increase of CD from 25 to 247.

Reference [31] sketches results of experiments for a 5-way case built from two 4-way cases. Since [31] does not provide a detailed description of the set of statements of the case, FS of the corresponding DNF formula cannot be determined. But extrapolating from the structure of the previous cases, we conclude that FS must still be small. Unfortunately, the average solution time is not provided, so CD cannot be computed. But an indication of the comprehension difficulty is the fact that the correctness rate k/n is 0.55, which is barely above the average performance of random choice. Thus, it seems fair to conclude that the 5-way case is close to or at the limit of human comprehension.

A potential criticism of the above encoding of sets of statements in formulas S is that it only implicitly accounts for the crucial concept of rating differences. But that objection would not be raised if the knowledge domain X could be easily restated using the same literals. Indeed, the main point of the example is the following. If the knowledge domain X isn't already expressed or cannot be easily restated in the terminology of the sets of statements or, equivalently, in terms of the literals of S, then chunking is difficult or impossible. Furthermore, in such cases the human brain has great difficulty processing more than just a few statements or formulas.

Despite the above examples, we would like to make some use of FS when estimating CD. We do this next.

2.4 Comprehensibility Condition

We say that a formula *directly uses* concepts of knowledge domain X if each literal directly corresponds to some concept of X. Suppose two formulas directly use concepts of X. In light of the preceding discussion and results, it is reasonable to assume that, in the absence of non-elementary chunking, the formula with smaller size *FS* is easier to understand. We summarize that assumption.

Monotonicity Assumption. Let knowledge domain X and formulas that directly use concepts of X be given, and suppose that non-elementary chunking is not possible. Then for any two formulas, the one with smaller *FS* value is easier to understand.

During the construction of a formula S, we do not know whether non-elementary chunking is possible when S is evaluated in light of a knowledge domain X. It seems prudent that we conservatively assume that such chunking is not possible. Under the Monotonicity Assumption, we then should strive for a formula S that directly uses concepts of X and has smallest size among the formulas with that feature. Indeed, in agreement with [32], we postulate the following condition for comprehensibility.

Comprehensibility Condition. Let knowledge domain X and a formula S that directly uses concepts of X be given. If *FS* is at most 4, then the formula S likely is comprehensible. On the other hand, if *FS* is larger than 4 and non-elementary chunking is not possible, then comprehensibility of the formula S is doubtful.

3 Construction of Comprehensible Formulas

This section summarizes a method for the construction of formulas that likely are comprehensible according to Section 2.4. The method uses SV (single-variable) and RHS (restricted-half-space) discretization. SV discretization has the customary form; one or more cutpoints are defined for each variable, and logic variables represent the occurrence of values in the various intervals. Specifically, for each cutpoint α of a variable x_j, we define a logic variable $w_{j,\alpha}$ that takes on the value *True* if $x_j > \alpha$, and the value *False* if $x_j < \alpha$. RHS discretization is an extension of SV discretization. Here, for a vector z of original variables, we select a cutpoint α and a vector b containing a few nonzero entries—shortly we make this notion precise—, and define a logic variable $w_{b,\alpha}$ to take on the value *True* if $b^t \cdot z > \alpha$, and the value *False* if $b^t \cdot z < \alpha$.

We assume that a learning machine $\mathcal{M}$ is given that derives classifying formulas from discretized data. The construction process uses SV and RHS discretization and learning machine $\mathcal{M}$ in a stagewise process. In stage 0, thelearning machine $\mathcal{M}$ receives training data discretized by SV discretization and computes a formula as output. Stages 1, 2, ... are like stage 0, except that SV discretization is replaced by a RHS discretization procedure that increases in complexity with the stage index. Thus, each stage produces an output formula.

For stage p, $p = 0, 1, \ldots$, denote by $\mathcal{M}_p$ the machine consisting of the relevant discretization procedure and the subsequently executed learning machine $\mathcal{M}$. As we shall see in Section 5, $\mathcal{M}_p$ may produce formulas whose literals $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$ have up to 2^p nonzero b_j. Since we want comprehensible formulas, we limit 2^p to 4 in agreement with Section 2.1, and thus enforce $p \leq 2$.

From the formulas obtained by the machines $\mathcal{M}_p$, $p \leq 2$, we want to select one that likely is best according to comprehensibility and reliability. In the long version of this paper, we describe how this can be done using the Monotonicity Assumption of Section 2.4 and the theory of VC dimension [48]. Here, we can only list the main conclusion: The selected formula should have high accuracy on the training data and small formula size.

The next section reviews SV discretization.

4 Review of SV Discretization

In the early developments of discretization methods, *entropy* combined with the *minimum description length principle* ([22, 23, 45]) and other schemes using information gain were the most widely used methods, with strong performance with regard to prediction accuracy; see for example [3, 4, 21, 35]. Recent research has produced a number of new discretization algorithms [12, 13, 37, 50, 44]. Using a *generalized entropy function*, [33] unifies and generalizes results. The above references cover research on *univariate discretization*, where each variable is treated by itself. In *multivariate discretization*, several variables are considered simultaneously, and a discretization of each variable is determined that takes into account the interaction of the variable with other variables. This more elaborate effort is undertaken with the expectation that the subsequently applied learning method will discover classifiers that in some sense are more meaningful. Representative references are [9, 14, 15, 26, 41, 43]. Formally, the output of multivariate discretization has the same structure as that of univariate discretization. That is, the output consists of cutpoints for the variables, and the variables are discretized according to these cutpoints.

A fundamental problem induced by discretization is the fact that near any cutpoint of a variable x_j a small random change of the x_j value may change the encoding. There are several ways to deal with this difficulty.

In the first approach, the discretization assigns several cutpoints to a given variable for a fine-grained representation. This solution has a hidden cost, in the form of more complex and thus less comprehensible formulas. In a second, fuzzy approach, membership functions are introduced that permit membership in several intervals; see for example [6]. Here, too, formulas can become complex and incomprehensible. A third approach introduces an uncertainty interval around each cutpoint; see for example [11], where the intervals are dynamically decided during the learning process. The disadvantage of the third approach is that the learning method must be able to process discretized data where some entries are marked "unknown."

RHS discretization relies on the third approach. Specifically, the width of the uncertainty interval enclosing a given cutpoint for a variable x_j should be large enough that random changes of the x_j value turn with low probability a value below the uncertainty interval into one above that interval, and *vice versa*. We define the *average uncertainty width* u_j of variable x_j to be the average of the widths of the uncertainty intervals enclosing the cutpoints of x_j.

5 RHS Discretization

We assume to have (1) an SV discretization method that outputs for each variable x_j besides the cutpoints an average uncertainty width u_j, and (2) a feature selection method (see [30, 36, 39]) that outputs the value of a reasonable importance measure. The RHS discretization process is as follows.

First, the scheme applies the assumed feature selection method to get importance values for the variables. Second, the scheme creates new variables that are linear combinations of variables. Third, the method uses SV discretization to obtain cutpoints for the new variables. Effectively, the cutpoints constitute thresholds of linear combinations of original variables.

We pause for a moment to point out that the use of linear combinations of variables followed by application of thresholds is central to well-known techniques of Machine Learning. For example, artificial neural nets (ANNs) use the two steps as fundamental building block. As a second example, support vector machines (SVMs) use the two steps in a typically high dimensional space into which the training data have been mapped.

While ANNs and SVMs in general allow linear combinations of any number of variables, here we use just pairs of variables. The rule for selecting the pairs depends on the stage of the overall process. For the moment, we skip discussion of that aspect and focus on the definition of new variables from one pair of variables, say involving x_k and x_l.

For $j = k, l$, suppose SV discretization has created just one cutpoint c_j for x_j, and let u_j denote the associated uncertainty width. Let R be the rectangle in the Euclidean plane that has the four points given by $(c_k \pm u_k/2, c_l \pm u_l/2)$ as corner points. The discretization rules implied by the cutpoints and their uncertainty intervals assign for any point in R the value "unknown" to both coordinates. By the derivation of the cutpoints and uncertainty intervals, that assignment is based on the behavior of x_k by itself and of x_l by itself. Maybe analysis of the interaction of x_k and x_l would support a more precise discretization where R is replaced by a smaller region. There are several ways to achieve this goal. Here, we opt for the subdivision of the rectangle R induced by its two diagonals. We explore the effect of that subdivision rather indirectly, as follows.

For some constants α_1 and α_2, the lines passing through the two diagonals of R are given by

$$\begin{aligned} x_k/u_k + x_l/u_l &= \alpha_1 \\ x_k/u_k - x_l/u_l &= \alpha_2 \end{aligned} \tag{2}$$

We could revise the discretization of x_k and x_l directly using these two lines. But more effective is the following approach, which momentarily considers α_1

and α_2 to be undecided, and which defines two variables y^+ and y^- by

$$\begin{aligned} y^+ &= x_k/u_k + x_l/u_l \\ y^- &= x_k/u_k - x_l/u_l \end{aligned} \tag{3}$$

We add these two variables as new attributes to the training data and compute training values for them by inserting the training values for x_k and x_l into (3). Due to this expansion of the space of attributes, we call the variables y^+ and y^- *expansion variables* and refer to the enlarged training data as *expanded training data*. The subsequent use of the expansion variables is decided by SV discretization applied to the expanded training data. That step decides cutpoints for the expansion variables that generally may make better use, so to speak, of the expansion variables than selection of α_1 and α_2 of (2) as cutpoints.

In the general case of several cutpoints for each of the variables x_j, we use the average uncertainty width as u_j in the definition (3) of y^+ and y^-.

In the construction process of Section 3, all variables on hand at the end of a stage, including all expansion variables created so far, are candidates for pairs producing additional expansion variables for the next stage. Since stage 0 carries out SV discretization, the variables available at the end of stage 0 are just the original variables. Thus, all expansion variables created in stage 1 are derived from original variables and, by (3), are defined by equations with two nonzero coefficients. Inductively, if the defining equations of the expansion variables created in stage $p \geq 1$ are rewritten in terms of the original variables, then the resulting equations have at most 2^p nonzero coefficients.

We restate this as follows. Let y be a new expansion variable of stage p, $p \geq 1$. Suppose the literal $(y < \alpha)$ or $(y > \alpha)$ occurs in a formula learned from the training data on hand in stage p. In terms of the vector x of original variables, the variable y is defined by $y = b^t \cdot x$, where b has at most 2^p nonzero entries, and any literal $(y > \alpha)$ or $(y < \alpha)$ can be rewritten as $(b^t \cdot x > \alpha)$ or $(b^t \cdot x < \alpha)$, respectively.

In Section 2.1, we constrained the literals $(b^t \cdot z > \alpha)$ or $(b^t \cdot z < \alpha)$ to b vectors with $k \leq 4$ nonzero b_j so that elementary chunking of the literal is possible. In the present setting, literals produced via RHS discretization observe that constraint if the number p of stages satisfies $2^p \leq 4$ and thus $p \leq 2$. From now on, we impose that limit on p.

Finally, we note that nominal attributes, where values are members of a nonnumerical set, do not participate in RHS discretization.

The next section discusses an implementation.

6 Implementation and Computational Results

We have added RHS discretization to an extension of the method of [46], which in turn is based on the discretization method of [7, 42] and the Lsquare algorithm of [24, 25]. We call the resulting method EXRHS (= EXplanations via RHS discretization). For present purposes, the details of EXRHS and the prior scheme of [46] it is based on, are not important. Suffice it to say that the prior scheme computes uncertainty intervals as part of SV discretization and establishes importance values of variables in a feature selection subroutine. As a result, the addition of RHS discretization is rather straightforward.

We use data sets of UC Irvine Machine Learning Repository for tests of EXRHS. The repository has a list of 11 most popular data set. Of these, 6 data sets are well suited for tests of RHS discretization since they involve no or almost no nominal attributes and allow easy formulation of two-class classification problems.

In order of popularity, the selected 6 data sets are Iris, Wine, Breast Cancer Wisconsin, Abalone, Yeast, and SPECT Heart. For each test, we split the given data set in a 50/50 ratio into training and testing data and then apply EXRHS. The 50/50 choice provides in all cases reasonable training and testing sets for evaluation of the effect of RHS discretization.

The Iris data set has a total of 150 records of three classes represented by 50 records each. Goal is to explain, for each class, the difference between that class and the remaining two classes. Thus, there are three cases: Iris-1 of class 1 versus classes 2, 3; Iris-2 of class 2 versus classes 1, 3; and Iris-3 of class 3 versus classes 1, 2. The Wine data set has a total of 178 records covering three classes of wines with 59, 71, and 48 records, respectively. The goal is to explain, for each class, the difference between that class and the remaining two classes. Thus, there are three cases Wine-1, Wine-2, and Wine-3 analogously to the Iris data set. The Breast Cancer Wisconsin data set has 569 records covering benign and malignant breast tumor cases. Goal is to explain the difference between the two diagnoses. The Abalone data set has a total of 4,177 records. Each record represents an abalone instance. Goal is to decide the number of rings of the instance using various measurements. The number of rings ranges from 1 to 29. In [16], a 3-class classification problem is defined for the Abalone data set analogously to the Iris data set, except that here class 1 has the records with 1-8 rings, class 2 has those with 9 or 10 rings, and class 3 contains those with 11-29 rings. The three tests cases are called Abalone-1, Abalone-2, and Abalone-3. The Yeast data set contains 1,484 records. The cellular localization sites of proteins are to be determined. There are ten classes. For the tests, we select the three most frequently occurring classes, called CYT, NUC, and MIT. For each case, the class is to be separated from the remaining cases. This produces three test cases Yeast-1, Yeast-2, and Yeast-3. The SPECT Heart data set has 267 records. In contrast to the above data sets, all attributes are binary. The goal is prediction of a $\{0, 1\}$ DIAGNOSIS variable. The original data set is given by training and testing sets of somewhat odd proportions. That is, the training data have 40 records each for the two DIAGNOSIS cases, but the testing data have just 15 cases for DIAGNOSIS $= 0$ and 172 cases for DIAGNOSIS $= 1$. For consistency with the above tests, we concatenate the original training and testing sets and then split the resulting set 50/50 into training and testing sets for use by EXRHS.

Looking over the data sets, it seems reasonable to claim that the attributes directly represent concepts of the underlying knowledge domain. Thus, it is also reasonable to invoke the Monotonicity Assumption and Comprehensibility Condition. Thus, formulas with size $FS \leq 4$ likely are comprehensible. On the other hand, if non-elementary chunking cannot be done, then comprehensibility of formulas with size $FS > 4$ is doubtful.

In all tests, EXRHS runs with the default rules and settings, and no selection or tuning of parameters is used. Suppose the output formulas of stages 0, 1, and 2 have been computed. We do not use application-dependent rules to select the final formula from the output formulas, since we have no guidance for selection of such rules. In agreement with the earlier discussion, we aim for a final formula that likely is comprehensible by eliminating all output formulas whose size exceeds 4, then select the formula with highest training accuracy from the remaining formulas. If case of a tie, the formula with lower stage index is chosen. In Table 6.1, the results for the final formulas are summarized in the three columns under the heading Best of Size ≤ 4; the column FS has the formula size, and the remaining two columns have the training and testing accuracy. We contrast the final formulas with the output formulas of Stage 0 in three analogous columns under the heading Stage 0 Output. These output formulas are constructed using SV discretization, and no restriction is imposed on their size.

Table 6.1 Summary for Formulas

	Best of Size ≤ 4			Stage 0 Output		
		Accuracy	(%)		Accuracy	(%)
Case	FS	Train	Test	FS	Train	Test
Iris-1	1	98	97	1	98	97
Iris-2	3	95	91	3	90	85
Iris-3	1	95	98	1	92	98
Wine-1	1	97	95	1	93	88
Wine-2	2	94	92	1	92	89
Wine-3	2	100	98	2	100	98
Breast	2	95	93	10	97	92
Abal.-1	2	80	79	5	81	79
Abal.-2	4	64	63	13	64	63
Abal.-3	1	70	71	6	72	72
Yeast-1	no size ≤ 4			17	61	60
Yeast-2	no size ≤ 4			10	68	64
Yeast-3	1	73	75	12	76	78
SPECT	2	81	72	2	81	72
Average	1.8	86.8	85.3	4.8	86.3	84.3

The results for Yeast-1 and Yeast-2 are not included in the averages

The columns under Best of Size ≤ 4 in Table 6.1 show that EXRHS finds likely-to-be-comprehensible explanations in 12 out of 14 (= 86%) of the cases.

The formula size *FS* is equal to 1 or 2 with two exceptions, where the size is 3 or 4. The average formula size is 1.8. It turns out that RHS discretization is used for 9 out of the 12 (= 75%) likely-to-be-comprehensible explanations. The 9 cases are split into 3 (= 33%) cases computed in stage 1, while the remaining 6 cases (= 67%) are determined in stage 2.

Of the 14 formulas listed under Stage 0 Output, only 7 cases (= 50%) have formula size ≤ 4 and are deemed likely comprehensible. Unless non-elementary chunking is possible, comprehensibility of the remaining 7 formulas is doubtful. The formula size *FS* ranges from 1 to 17. The average formula size is 4.8 if the formulas produced by Yeast-1 and Yeast-2, for which Best of Size ≤ 4 has no counterpart, are ignored.

The accuracy averages at the bottom of Table 6.1 show that the improvement in comprehensibility due to RHS discretization is accompanied by a small average training accuracy gain of $86.8 - 86.3 = 0.5\%$ and a small average testing accuracy gain of $85.3 - 84.3 = 1.0\%$. Thus, RHS discretization produces not only likely-to-be-comprehensible formulas for 86% of the cases, but does so with a small training and testing accuracy gain rather than a loss.

Test results reported for prior schemes that compute logic formulas often cite the number of computed rules. Since the formula size of a rule is at least 1, the number of rules of an explanation is a lower bound on the formula size of the explanation. That lower bound can be quite large. For example, some papers list 12, 26, 19, 42, 77, 27-109, average 18, average 6, average 16, ≥ 18 rules for two-class classification problems handled by various methods. Maybe RHS discretization can help to reduce those numbers.

Table 6.2 provides total execution times for EXRHS in min:sec, based on runs on a 3.0GHz PC. When three cases are listed together on a line, the stated execution time is the total run time for those cases. In addition, column Attr. gives the number of attributes, not counting the class attribute, and column Train Recds has the number of training records.

Table 6.2 Execution Times of EXRHS

Case	Attr.	Train Recds	Exec. Time (min:sec)
Iris(1-3)	4	75	0:01
Wine(1-3)	13	89	0:02
Breast	30	285	0:03
Abal.(1-3)	8	2,089	1:57
Yeast(1-3)	8	742	0:18
SPECT	22	134	0:01

7 Extension

EXRHS has been extended to carry out Subgroup Discovery. The scheme is somewhat different from typical current Subgroup Discovery approaches [34, 49], which often use an iterative investigation involving the steps of data mining, interpretation of results, and evaluation by experts; see for example [5, 27–29, 38]. In contrast, EXRHS typically determines a few highly significant subgroups without manual guidance. Each subgroup is characterized by a convex polyhedron that generally isn't just an axis-parallel rectangle.

8 Summary

Using prior studies in Neurology and Psychology, the paper links comprehensibility of explanations given by logic formulas to the size of the formulas. That is, if a formula is not too large in a certain sense, then under certain assumptions we can expect it to be understood by humans. On the other hand, if formula size exceeds a certain bound, then under certain assumptions it may well be the case that the formula cannot be comprehended by humans.

The paper introduces a restricted form of discretization called RHS discretization that can be computed with the aid of any number of prior discretization and feature selection methods. In computational tests using well-known data sets, it is shown that RHS discretization produces likely-to-be-comprehensible explanations for 86% of the cases. In contrast, when the traditional discretization approach via cutpoints is used, only 50% of the explanations have that feature. The improvement of comprehensibility is accompanied by a small average gain in accuracy instead of a loss.

The methodology has been extended to cover Subgroup Discovery.

References

1. S. Abidi and K. Hoe. Symbolic exposition of medical data-sets: A data mining workbench to inductively derive data-defining symbolic rules. In *Proceedings of the 15th IEEE Symposium on Computer-based Medical Systems (CBMS'02)*, 2002.
2. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993.
3. A. An. Learning classification rules from data. *Computers and Mathematics with Applications*, 45:737–748, 2003.
4. A. An and N. Cercone. Discretization of continuous attributes for learning classification rules. In *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, 1999.
5. M. Atzmueller, F. Puppe, and H.-P. Buscher. Subgroup mining for interactive knowledge refinement. In *Proceedings of the 10th Conference on Artificial Intelligence in Medicine (AIME 05)*, 2005.
6. W.-H. Au, K. C. C. Chan, and A. K. C. Wong. A fuzzy approach to partitioning continuous attributes for classification. *IEEE Transactions on Knowledge and Data Engineering*, 18:715–719, 2006.
7. S. Bartnikowski, M. Granberry, J. Mugan, and K. Truemper. Transformation of rational and set data to logic data. In *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques.* Springer, 2006.

8. S. Bay and M. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5:213–246, 2001.
9. S. D. Bay. Multivariate discretization of continuous variables for set mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.
10. E. Boros, P. Hammer, T. Ibaraki, and A. Kogan. A logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.
11. E. Boros, P. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12:292–306, 2000.
12. M. Boullé. Khiops: A statistical discretization method of continuous attributes. *Machine Learning*, 55:53–69, 2004.
13. M. Boullé. MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65:131–165, 2006.
14. S. Chao and Y. Li. Multivariate interdependent discretization for continuous attribute. In *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, 2005.
15. M. R. Chmielewski and J. W. Grzymala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, 15:319–331, 1996.
16. D. Clark, Z. Schreter, and A. Adams. A quantitative comparison of dystal and backpropagation. In *Proceedings of Seventh Australian Conference on Neural Networks (ACNN'96)*, 1996.
17. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings Fifth European Working Session on Learning*, 1991.
18. W. W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.
19. W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.
20. N. Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185, 2001.
21. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.
22. U. Fayyad and K. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.
23. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.
24. G. Felici, F. Sun, and K. Truemper. Learning logic formulas and related error distributions. In *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*. Springer, 2006.
25. G. Felici and K. Truemper. A MINSAT approach for learning in logic domain. *INFORMS Journal of Computing*, 14:20–36, 2002.
26. N. Friedman and M. Goldszmidt. Discretizing continuous attributes while learning Bayesian networks. In *International Conference on Machine Learning*, 1996.
27. D. Gamberger and N. Lavrač. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.
28. D. Gamberger, N. Lavrač, and G. Krstačic. Active subgroup mining: a case study in coronary heart disease risk group detection. *Artificial Intelligence in Medicine*, 28, 2003.

29. D. Gamberger, N. Lavrač, F. Železný, and J. Tolar. Induction of comprehensible models for gene expression datasets by subgroup discovery methodology. *Journal of Biomedical Informatics*, 37, 2004.
30. I. Guyon and A. Elisseef. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
31. G. S. Halford, R. Baker, J. E. McCredden, and J. D. Bain. How many variables can humans process? *Psychological Science*, 16:70–76, 2005.
32. G. S. Halford, N. Cowan, and G. Andrews. Separating cognitive capacity from knowledge: a new hypothesis. *Trends in Cognitive Sciences*, 11:236–242, 2007.
33. R. Jin, Y. Breitbart, and C. Muoh. Data discretization unification. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'07)*, 2007.
34. W. Klösgen. EXPLORA: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.
35. R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.
36. D. Koller and M. Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, 1996.
37. L. A. Kurgan and K. J. Cios. CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16:145–153, 2004.
38. N. Lavrač, B. Cestnik, D. Gamberger, and P. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning*, 57:115–143, 2004.
39. H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17:491–502, 2005.
40. G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.
41. S. Monti and G. F. Cooper. A multivariate discretization method for learning Bayesian networks from mixed data. In *Proceedings of the Fourteenth Conference of Uncertainty in AI*, 1998.
42. J. Mugan and K. Truemper. Discretization of rational data. In *Proceedings of MML2004 (Mathematical Methods for Learning)*. IGI Publishing Group, 2007.
43. F. Muhlenbach and R. Rakotomalala. Multivariate supervised discretization, a neighborhood graph approach. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'02)*, 2002.
44. P. Perner and S. Trautzsch. Multi-interval discretization for decision tree learning. In *Advances in Pattern Recognition*. Springer Verlag, 2004.
45. J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
46. K. Riehl. *Data Mining Logic Explanations from Numerical Data*. PhD thesis, Department of Computer Science, University of Texas at Dallas, 2006.
47. E. Triantaphyllou. *Data Mining and Knowledge Discovery via a Novel Logic-based Approach*. Springer, 2008.
48. V. Vapnik, E. Levin, and Y. L. Cun. Measuring the VC-dimension of a learning machine. *International Journal of Human Computer Systems*, 6:851–876, 2008.
49. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of First European Conference on Principles of Data Mining and Knowledge Discovery*, 1997.
50. Y. Yang and G. I. Webb. Weighted proportional k-interval discretization for Naive-Bayes classifiers. In *Proceedings of the Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-03)*, 2003.