Data Mining Using Multi-Valued Logic Minimization

Tsutomu Sasao
Department of Computer Science, Meiji University, Kanagawa, Japan

***Abstract*—In a partially defined classification function, each input combination represents features of an example, while the output represents the class of the example. Each variable may have different radix. In this paper, we show a method to minimize the number of variables. Combined with a multiple-valued logic minimizer, data sets of examples are represented by a compact set of rules. Experimental results using University of California Irvine (UCI) benchmark functions show the effectiveness of the approach, especially for imbalanced data sets. The results are compared with J48 and JRIP. This approach produces explainable 100% correct rules, which are promising for bio-medical applications.**

***Index Terms*—multi-valued logic, partially defined function, classification, decision tree, imbalanced data set, variable minimization**

I. Introduction

Data mining is a process of discovering knowledge in sample data sets. Compact representation of large data enables us to find knowledge in the data set [1]. In many cases, the size of the sample data set is much smaller than the all possible input combinations. In such a case, the number of variables can be often reduced. By reducing the number of variables, and by simplifying the multiple-valued expression, we can derive a compact set of rules. These rules are explainable [7], [10] . In this paper, we show a method to reduce the number of input variables of such functions. The method generates rules consistent with the sample data set. For University of California Irvine (UCI) benchmark functions, we show that many variables can be reduced, and compact set of rules can be generated.

Contributions of this paper are

- Presented a new method to derive rules for a given set of examples.
- The accuracy of the generated rules is 100%.
- Compared with C4.5 [15] (a standard decision tree algorithm), and JRIP [6] , and showed the usefulness of the approach, especially for imbalanced data sets.

The rest of the paper is organized as follows: Section II shows definitions and basic properties of classification functions; Section III shows a method to minimize the number of variables; Section IV introduces words in data mining and logic minimization; Section V shows experimental results for standard UCI data sets; Section VI shows experimental results for imbalanced data sets; Section VII compares results with existing methods; and Section VIII concludes the paper.

TABLE 2.1
Registered Vector Table

	X_1	X_2	X_3	X_4	f	
$\vec{a}_1$	2	3	3	2	1	F_1
$\vec{a}_2$	1	3	2	3	1	
$\vec{b}_1$	3	2	2	1	2	F_2
$\vec{b}_2$	3	1	2	1	2	
$\vec{c}_1$	2	3	2	1	3	F_3
$\vec{c}_2$	1	3	2	1	3	

II. Definitions and Basic Properties [18]

Definition 2.1: Let $n \geq 1$ and, for each $1 \leq i \leq n$, $P_i = \{1, 2, \ldots, p_i\}$, where $p_i \geq 1$. Let P be the Cartesian product composed of $P_1, P_2, \ldots, P_n$, i.e., $P = P_1 \times P_2 \times \cdots \times P_n$, and D be a non-empty subset of P, i.e., $\varnothing \neq D \subset P$. Let $k = |D|$ and $M = \{1, 2, \ldots, m\}$ for some m satisfying $2 \leq m \leq k$. An element of D is called a **registered vector**. A mapping $f : D \rightarrow M$ is a **classification function**. A value of f is called a **class**.

We sometimes write $F_i = f^{-1}(i), i \in M$, when f is understood. It is clear that $D = \bigcup_{i=1}^{m} F_i$. We assume that $F_i \neq \varnothing$ for every $i \in M$.

Example 2.1: The registered vector table in Table 2.1 shows a classification function with $n = 4$, $m = 3$ and $k = 6$. ∎

Let D, M and m be given in Definition 2.1. Suppose that a sequence $\langle F_1, F_2, \ldots, F_m \rangle$, where $F_i \subset D$ for each $i \in M$, is a partition of D, that is $D = \bigcup_{i=1}^{m} F_i$ and $F_i \cap F_j = \varnothing$ for $i, i \in M$ with $i \neq j$. Then the function $f : D \rightarrow M$ such that $F_i = f^{-1}(i)$ for $i \in M$ will be denoted by $(F_1, F_2, \ldots, F_m)$.

For the next definition, two functions with different domains are considered. So, one has to be careful about determining m and, hence M.

Definition 2.2: For classification functions $f : D \rightarrow M$ and $g : E \rightarrow M$ with the same range $M = \{1, 2, \ldots, m\}$ where $2 \leq m \leq \min\{|D|, |E|\}$, g is an **extension** of f if $f^{-1}(i) \subseteq g^{-1}(i)$ for all $i \in M$.

Definition 2.3: [2] For a subset $U \subseteq D$ and $S \subseteq \{1, 2, \ldots, n\}$, we denote by $U|_S$ the **projection** of U to S. In other words, $U|_S$ is obtained from U by considering only the j-th components, where $j \in S$.

Example 2.2: Let $U = \{(1, 2, 3, 1), (3, 1, 1, 2), (2, 3, 1, 2)\}$ and $S = \{2, 3\}$. Then, we have the projection $U|_S = \{(*, 2, 3, *), (*, 1, 1, *), (*, 3, 1, *)\}$. ∎

TABLE 2.2
REDUCED REGISTERED VECTOR TABLE

X_2	X_4	f
3	2	1
3	3	1
2	1	2
1	1	2
3	1	3
3	1	3

Given a partially defined function, many extensions exist. In this paper, we seek an extension of f that depends on the fewest variables.

Definition 2.4: Let $F_i \subset D$ $(i = 1, 2, \ldots, m)$. Given a classification function $(F_1, F_2, \ldots, F_m)$ of n variables, and a subset $S \subseteq \{1, 2, \ldots, n\}$, if $F_i|_S \cap F_j|_S = \varnothing$, $(i \neq j)$ holds, then S is a **support set**. In such a case, $(F_1|_S, F_2|_S, \ldots, F_m|_S)$ is independent of the variable X_j, where $j \in \{1, 2, \ldots, n\} \setminus S$. In this case, X_j is **redundant**.

Example 2.3: Consider the function (F_1, F_2, F_3) shown in Table 2.1. In this case, $S = \{2, 4\}$ is a support set, since for

$$\begin{aligned} F_1|_S &= \{(*, 3, *, 2), (*, 3, *, 3)\}, \\ F_2|_S &= \{(*, 2, *, 1), (*, 1, *, 1)\}, \\ F_3|_S &= \{(*, 3, *, 1), (*, 3, *, 1)\}, \end{aligned}$$

$F_i|_S \cap F_j|_S = \varnothing$ holds for $i < j$. Thus, this function can be represented by two variables, as shown in Table 2.2. ∎

III. MINIMIZATION OF VARIABLES

In this part, we show a method to minimize the number of variables in the support set.

Definition 3.1: Let $f : D \to M$ be a classification function with n variables. If there exist $\vec{a} = (a_1, a_2, \ldots, a_n)$, $\vec{b} = (b_1, b_2, \ldots, b_n) \in D$ and $i \in \{1, 2, \ldots, n\}$ such that $a_j = b_j$ for any $j \in \{1, 2, \ldots, n\} \setminus \{i\}$, $a_i \neq b_i$ and $f(\vec{a}) \neq f(\vec{b})$, then f is said to **depend** on the i-th variable.

Theorem 3.1: If a function f depends on x_i, any support set of f contains i.

(Proof) If a support set does not contain i, then there are no vectors $\vec{a}$ and $\vec{b}$ such that $f(\vec{a}) \neq f(\vec{b})$ and $\vec{e}_i = \vec{a} \oplus \vec{b}$. This means that f is independent of x_i. □

When a function f depends on x_i, then x_i is said to be **essential**, and any SOP for f requires x_i.

Example 3.1: Consider the function shown in Table 2.1. It depends on X_4, since

$$\begin{aligned} &\vec{a}_2 = (1, 3, 2, 3) \in F_1 \quad \text{and} \\ &\vec{c}_2 = (1, 3, 2, 1) \in F_3. \end{aligned}$$

Thus, X_4 is essential. ∎

The following algorithm is an extension of two-valued cases to multi-valued cases.

Algorithm 3.1: (Minimization of Variables)

1) For each pair $(\vec{a}, \vec{b})$ such that $\vec{a} \in F_\alpha$ and $\vec{b} \in F_\beta$, $(\alpha \neq \beta)$ make a clause

$$C(\vec{a}, \vec{b}) = z_1 \vee z_2 \vee \cdots \vee z_n,$$

where

$$z_j = \begin{cases} 0 & \text{if} \quad a_j = b_j \\ y_i & \text{if} \quad a_j \neq b_j. \end{cases}$$

2) For all the pairs $(\vec{a}, \vec{b})$ in $\vec{a} \in F_\alpha$ and $\vec{b} \in F_\beta$, construct the product of the clauses.

$$R = \bigwedge_{(\vec{a}, \vec{b})} C(\vec{a}, \vec{b}).$$

3) Covert the expression of R into a sum-of-products, and simplify it. A product with the fewest literals corresponds to a minimum support set.

Example 3.2: Consider the function in Table 2.1.

1) The set of clauses are:

$$\begin{array}{ll} C(\vec{a}_1, \vec{b}_1) = y_1 \vee y_2 \vee y_3 \vee y_4, & C(\vec{a}_1, \vec{b}_2) = y_2 \vee y_3 \vee y_4, \\ C(\vec{a}_1, \vec{c}_1) = y_3 \vee y_4, & C(\vec{a}_1, \vec{c}_2) = y_1 \vee y_3 \vee y_4, \\ C(\vec{a}_2, \vec{b}_1) = y_1 \vee y_2 \vee y_4, & C(\vec{a}_2, \vec{b}_2) = y_1 \vee y_2 \vee y_4, \\ C(\vec{a}_2, \vec{c}_1) = y_1 \vee y_4, & C(\vec{a}_2, \vec{c}_2) = y_4, \\ C(\vec{b}_1, \vec{c}_1) = y_1 \vee y_2, & C(\vec{b}_1, \vec{c}_2) = y_1 \vee y_2, \\ C(\vec{b}_2, \vec{c}_1) = y_1 \vee y_2, & C(\vec{b}_2, \vec{c}_2) = y_1 \vee y_2, \end{array}$$

2) Forming the product of all the clauses yields:

$$\begin{aligned} R \;=\; & (y_1 \vee y_2 \vee y_3 \vee y_4)(y_2 \vee y_3 \vee y_4)(y_3 \vee y_4) \\ & (y_1 \vee y_3 \vee y_4)(y_1 \vee y_2 \vee y_4)(y_1 \vee y_4) y_4 (y_1 \vee y_2). \end{aligned}$$

3) The simplified expression is

$$R = (y_1 \vee y_2) y_4 = y_1 y_4 \vee y_2 y_4.$$

The minimum support set is $\{1, 4\}$ and $\{2, 4\}$. ∎

IV. DATA MINING AND LOGIC DESIGN

In this part, we show relations between data mining and logic minimization. Note that the same notion is often called differently in different specializations. Table 4.1 shows the relations of words used in two specializations.

TABLE 4.1
TERMINOLOGY IN DIFFERENT AREAS OF SPECIALIZATION.

Logic minimization	Data mining
minterm	example, instance, sample
implicant	rule
SOP	covering rule set
variable	feature, attribute

In the data mining, each row in Table 2.1 corresponds to an example, an instance, or a sample.

Data mining is to find a simple representation of the sample set.

Definition 4.1: A set of rules is **complete** if it covers all the examples in each class. A set of rules is **consistent** if each rule covers examples in only one class, and none of examples in multiple classes.

Definition 4.2: Two examples are **inconsistent** or **conflicting** if they have the same input parts, but belong to different classes.

Minimization of the number of rules corresponds to minimization of the number of products in a sum-of-products expression (SOP). In logic design, specifications are often given by the ON sets and the DC (*don't care*) sets. On the other hand, in data mining, specifications are given by examples. Combinations not shown by the examples are undefined.

Example 4.1: Consider the function in Table 2.2. By a minimization of SOP of multiple-valued inputs, we have the following:

$$F_1 = X_4^{\{2,3\}}, \ F_2 = X_2^{\{1,2\}} \cdot X_4^{\{1\}}, \ F_3 = X_2^{\{3\}} \cdot X_4^{\{1\}}.$$

When English is used to represent the function, we have:

1) If $(X_4 = 2 \ OR \ 3)$, then $f = 1$, else
2) if $(X_2 = 1 \ OR \ 2)$ AND $(X_4 = 1)$, then $f = 2$, else
3) if $(X_2 = 3)$ AND $(X_4 = 1)$, then $f = 3$.

For the function in Table 2.1, the output values for only 6 combinations are specified, while the number of possible input combinations is $3^4 = 81$. Thus, the values for other $81 - 6 = 75$ combinations are undefined. On the other hand, in Table 2.2, each row shows $3^2 = 9$ combinations, and the last two rows are the same. Thus, Table 2.2 shows $5 \times 9 = 45$ combinations. For example, for the vector $(X_1, X_2, X_3, X_4) = (0, 3, 0, 2)$, the output is undefined in Table 2.1. However, in Table 2.2, the output is specified to $f = 1$. Fig. 4.1 show the map of the rules. In this case, value 1 are assigned to four blank cells. Note that these four combinations are undefined in Table 2.2. The logic minimizer assigned the value 1 to these cells. Thus, the rules have **generalization ability** for unseen input combinations.

However, it may be possible to have a map as shown in Fig. 4.2. In this case, value 2 are assigned to four blank cells. Note that the logic minimizer does not assign value 3 to these blank cells. ■

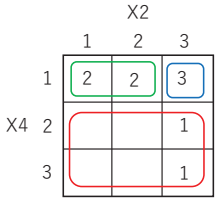

Fig. 4.1. Value 1 are assigned

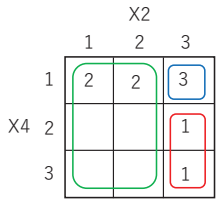

Fig. 4.2. Value 2 are assigned

V. Experiments with Standard Benchmarks

To show the effectiveness of the approach, we derived rules for selected benchmark functions from the UCI (University of California, Irvine) machine learning repository [22].

Table 5.1 summarizes the experimental results. The first column shows the name of the function. The second column shows the number of original variables: n. The third column shows the number of instances: k. The fourth column shows the number of classes: m. The fifth column shows the number of variables after variable minimization: n_e. The sixth column shows the number of rules after SOP minimization: r_e. Rules are generated for m classes. For example, in the case of *Monks* ($m = 2$), rules for both F_1 and F_2 were generated. However, in many cases, only one of two is sufficient. The seventh column shows the **rule complexity**: $\alpha = n_e \cdot r_e$. The eighth column shows n_t, the number of variables obtained by J48 [1]. The ninth column shows r_t, the number of rules obtained by J48. The tenth column shows the rule complexity of J48: $\beta = n_t \cdot r_t$. The eleventh column shows accuracy for J48. The last four columns show the same things for JRIP. To minimize the number of variables, Algorithm 3.1 was used, while to minimize SOPs with multi-valued inputs, heuristic algorithm MINI10 in [20] was used.

A. Breast Cancer

This function classifies the patients into two classes: (1) malignant, and (2) benign. The original dataset consists of 699 instances. We removed 16 incomplete instances (i.e., instances with missing entries), and six conflicting instances. Each variable takes 10 values.

B. Chess3196

This function shows whether the white can win or not for $k = 3196$ starting positions. We assume that each player plays optimally. Variables take 2, 3, or 4 values.

C. Connect-4

This function classifies the positions of the game into three: (1) player X wins, (2) player Y wins, or (3) draw. We assume that each player plays optimally. Each variable takes 3 values: empty, X, or Y.

D. Dermatology

This function classifies the patients into six classes. 10 conflicting instances were removed from the original data. Variables take 2, 3, 4, or 76 values.

E. Letter Recognition

This function classifies the image data of English alphabets into 26 classes. Each variable takes 16 values.

F. Monks

This function shows the relation of input variables. There are 6 variables: two of them take 2 values; three of them take 3 values; and one of them takes 4 values. The number of instances is $k = 2^2 \times 3^3 \times 4 = 432$. Note that this function is **totally defined**.

G. Mushroom

This function classifies the mushrooms into two classes: (1) edible, and (2) not edible. Originally, it contained 8124 instances, but 2480 incomplete instances were removed. Note that [12] shows a solution with five rules which depend on 8 variables, while our method produced a solution with five rules which depend on only 3 variables.

[1] J48 is a Java implementation of C4.5 on WEKA [23]. C4.5 is a standard decision tree classifier [24].

TABLE 5.1
RESULT FOR STANDARD BENCHMARK SETS.

	Orig. Var. n	Orig. Inst. k	Class m	Our Classifier Var. n_e	Rule r_e	Comp. α	J48 Var. n_t	Rule r_t	Comp. β	Acc. %	JRIP Var. n_t	Rule r_t	Comp. γ	Acc. %
Breast Cancer	9	677	2	4	15	60	7	12	84	99.0	7	6	42	98.5
Chess3196	36	3196	2	29	41	1189	22	30	660	99.7	22	17	374	99.6
Connect-4	42	67557	3	34	8356	284104	42	3871	162582	86.9	33	82	2706	75.5
Dermatology	34	358	6	6	53	318	7	8	56	98.0	10	8	80	96.9
Letter Recog	16	20000	26	11	1394	15334	16	1226	19616	96.3	16	428	6848	93.6
Monks	6	432	2	3	7	21	1	2	2	75.0	1	2	2	75.0
Mushrooms	22	5644	2	3	5	15	6	8	48	100.0	6	6	36	100.0
Promoter	57	106	2	4	10	40	7	10	70	98.1	7	4	28	93.4
Splice	60	3174	3	10	544	5440	41	101	4141	98.4	29	13	377	96.0
Teaching Assis	5	143	3	3	18	54	5	35	175	88.8	3	4	12	58.0
Tic Tac Toe	9	958	2	8	62	496	9	49	441	93.9	9	11	99	98.7
Vote	16	435	2	9	23	207	5	7	35	97.5	1	2	2	94.5
Zoo	16	101	7	5	9	45	8	9	72	99.0	8	7	56	94.1

H. Promoter

This function classifies the gene sequences (DNA) into two classes: (1) positive or (2) negative. Each variable takes 4 values: (1) A, (2) G, (3) T, and (4) C.

I. Splice

This function classifies the gene sequences (DNA) into three classes: (1) donor, (2) acceptor, (3) neither. Originally, it contained 3191 instances, but 16 ambiguous instances were removed. Each variable takes 4 values: (1) A, (2) G, (3) T, and (4) C.

J. Teaching Assistant Evaluation

This function classifies the performance of teaching assistants into three classes (1) low, (2) medium, and (3) high. Originally, it contained 151 instances, but 8 conflicting instances were removed. Variables take 2, 23 or 42 values.

K. Tic-Tac-Toe

This function classifies the patterns of the marking in a 3×3 grid into two classes: (1) player X wins, and (2) player X does not win. We assume that each player plays optimally. Each variable takes 3 values.

L. Vote

This function classifies the members of U.S. House of Representatives into two classes: (1) democrat, and (2) republican. Each variable takes 3 values.

M. Zoo

This function classifies 101 animals into 7 classes: (1) Mammals, (2) Birds, (3) Reptiles, (4) Fish, (5) Amphibians, (6) Insects, (7) Others.

Among 16 variables, X_{13} shows the number of legs and takes 6 values. Other variables take 2 values. Two variables X_6 and X_{13} are essential. There are seven minimal sets consisting of five variables.

VI. EXPERIMENTS WITH IMBALANCED DATA SET

When the number of instances in one class is much larger than the number of instances in the other class, the set is **imbalanced** [3]. Many practical classification problems have imbalanced class distributions.

In this part, we consider experimental results for imbalanced data sets. Table 6.1 shows the results. The meanings of the columns are the same as that of Table 5.1. In this part, we use the following data sets.

A. Hepatitis

This function classifies the patients into two classes: (1) death and (2) survival. Incomplete instances were removed. The number of instances in Class 1 and Class 2 are 13 and 67, respectively. Variables take 2, 5, 6, 7, or 11 values.

B. Thoracic

This data is related to the post-operative life expectancy in the lung cancer patients. The number of instances is 470. Among them, Class 1 (death within one year after surgery) consists of 70 instances, while Class 2 (survival) consists of 400 instances. The number of variables is $n = 16$: three of them are numerical, one takes 3, one takes 4, one takes 7 values, and ten of them take 2 values.

VII. COMPARISON WITH OTHER CLASSIFIERS

More than 179 classifiers have been developed [8]. Among them, C4.5 [15], a classifier based on decision trees, is well known [9], [24]. Thus, we compare our method with *J48*, a Java implementation of C4.5 on WEKA [23]. *JRIP* is a rule learner based on the RIPPER (Repeatedly Incremental Pruning to Produce Error Reduction) algorithm [6], which is similar to our approach. Thus, we also compare with JRIP.

A. Comparison of Goals

The goals of these classifiers are different. The goal of our classifier is to derive a compact set of rules that is consistent with the training set. Thus, the accuracy for the training set is 100%. On the other hand, the goals of C4.5 and JRIP are to derive a compact set of rules with high accuracy for both the training set and the test set. That is, to derive the rules

TABLE 6.1
RESULTS FOR IMBALANCED DATA SETS.

	Orig. Var. n	Orig. Inst. k	Class m	Our Classifier Var. n_e	Rule r_e	Comp. α	J48 Var. n_t	Rule r_t	Comp. β	MCC	JRIP Var. n_t	Rule r_t	Comp. γ	MCC
Hepatitis	19	80	2	4	8	32	3	4	12	0.703	2	2	4	0.689
Thoracic	16	470	2	4	28	112	0	1	0	UD	0	1	0	UD

with high **generalization ability**. In other words, our goal is **logic synthesis** for the training set, while the goals of C4.5 and JRIP are **approximate logic synthesis** for the training set.

Our classifier not only memorizes the training set, but also analyzes the data and generalizes them by

- Reduction of variables, and
- Assignment of values to *don't care* elements, which is done during SOP minimization. (Examples are shown in Figs. 4.1 and 4.2.)

C4.5 obtains its generalization ability by

- Pruning the tree, and
- Limiting the depth of the tree.

B. Comparison of Solutions

In the experiment, for each benchmark data, all the examples were used to train the decision trees. The parameters for J48 and JRIP were set to the default values of WEKA [23].

We assume that a solution with a smaller **rule complexity** is simpler. We also assume that the number of leaves of the tree is equal to the number of rules, because a path from the root node to a leaf corresponds to a rule.

To evaluate the performance of classifiers, we use the **confusion matrix** shown in Table 7.1, where TP denotes **true positive**, FP denotes **false positive**, FN denotes **false negative**, and TN denotes **true negative**.

TABLE 7.1
CONFUSION MATRIX

	Predicted	
Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

To show the performance of a classifier for imbalanced data sets, we use *Matthews Correlation Coefficient* (MCC) [5]. which is given by

$$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$$

When $FN = FP = 0$, $MCC = 1.00$, while when $TN = TP = 0$, $MCC = -1.00$. For the classifier that classifies all the instances to *Negative* ($TP = FP = 0$), or all the instances to *Positive* ($TN = FN = 0$), MCC is undefined (UD). In this case, the denominator of MCC is zero.

Analysis of Table 5.1

For *Monks*, J48 and JRIP produced poor solutions, i.e., the accuracy was 75%, although the sizes of their trees were small. Among 15 benchmark functions, only this function was totally defined. That is, the values of this function are specified for all possible input combinations.

For *Connect-4*, J48 and JRIP produced poor solutions, i.e., the accuracy was 86.9%, and 75.5%, respectively, while our classifier produced 100% accuracy. However, the rule complexities for J48 and JRIP were smaller.

For *Mushrooms*, three classifiers produced solutions with 100% accuracy, while our classifier produced the simplest solution.

In general, J48 produced solutions with higher accuracy than JRIP. Our classifier always produced solutions with 100% accuracy.

Analysis of Table 6.1

Here, we analyze the results for imbalanced data sets.

For *hepatitis*, our classifier produced 8 rules with four variables, and with MCC=1.00, while J48 produced decision tree with three variables, and four rules, and with MCC=0.793. JRIP produced two rule with two variables and MCC=0.689.

For *Thoracic*, our classifier produced 28 rules with four variables and MCC=1.00, while J48 produced a decision tree with only one leaf, and classified all the data into Class 2 (survival). In this case, MCC=UD. Thus, J48 is useless for this data set. JRIP produced the same solution.

C. Improvement by SMOTE and Cost-Sensitive Method

In general, straightforward applications of conventional classifiers to imbalanced sets results in poor solutions. Thus, various methods have been developed [13]. Among them, *SMOTE* (Synthetic minority over-sampling technique) [4] creates artificial data using an interpolation strategy that its k minority class nearest neighbors [3].

On the other hand, in the *Cost-Sensitive Method* [14], penalty of the error for the minor class is increased.

Example 7.1: Consider the case of *hepatitis*. The confusion matrices of our classifier, J48, and JRIP are shown in the upper and the middle rows. The confusion matrices of J48 and JRIP after applying SMOTE are shown in the bottom row.

$$M_{our} = \begin{bmatrix} 13 & 0 \\ 0 & 67 \end{bmatrix}$$

$$M_{J48} = \begin{bmatrix} 7 & 6 \\ 0 & 67 \end{bmatrix}, \quad M_{JRIP} = \begin{bmatrix} 10 & 3 \\ 4 & 63 \end{bmatrix}$$

$$M_{SJ48} = \begin{bmatrix} 63 & 2 \\ 1 & 66 \end{bmatrix}, \quad M_{SJRIP} = \begin{bmatrix} 64 & 1 \\ 8 & 59 \end{bmatrix}$$

In the original data, *MCC* by J48 and JRIP are 0.703 and 0.689, respectively.

TABLE 7.2
EFFECT OF SMOTE AND COST-SENSITIVE APPROACH FOR IMBALANCED DATA SETS.

	J48				JRIP				Method
	var	rule	Comp	MCC	var	rule	Comp	MCC	
Hepatitis	6	8	48	0.955	4	4	16	0.869	SMOTE4x
Hepatitis	7	8	56	0.881	5	4	20	0.848	Cost5x
Thoracic	12	55	660	0.741	10	12	120	0.666	SMOTE6x
Thoracic	14	61	854	0.684	11	18	198	0.501	Cost5x

Table 7.2 shows the results after applying SMOTE and Cost-Sensitive Methods. Note that SMOTE augmented 42 artificial positive data to improve the classifier, and improved *MCC* for J48 and JRIP to 0.955 and 0.869, respectively. On the other hand, the cost-sensitive approach improved the *MCC* for J48 and JRIP to 0.881 and 0.848, respectively. Note that our method produced MCC with 1.00, while the complexity of the rules are small. ■

VIII. CONCLUSION AND COMMENTS

This paper showed a method to reduce the number of variables for partially defined classification functions. With this strategy, numbers of variables for various functions were reduced successfully. Also, a multiple-valued SOP minimizer was used to reduce the number of rules. This method produces set of rules with 100% accuracy for given set of examples. It also compared **rule complexity** with C4.5 (J48) and JRIP. This method is useful for data mining, especially for bio-medical applications, since the obtained rules are simple and explainable. In bio-medical applications, the number of the samples are relatively small, and the data are often imbalanced [21], while rules with very high accuracy are required.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. The author thanks Prof. Jon T. Butler for discussion, and also thank the referee whose comments improved definitons.

REFERENCES

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Information Processing Letters*, Vol. 24, Issue 6, pp. 377-380,1987.

[2] E. Boros, T. Horiyama, T. Ibaraki, K. Makino, and M. Yagiura, "Finding essential attributes from binary data," *Annals of Mathematics and Artificial Intelligence,* Vol. 39, No. 3, pp. 223-257, Nov. 2003.

[3] P. Branco, L. Torgo, and R. P. Ribeiro, " A survey of predictive modeling on imbalanced domains," *ACM Comput. Surv.* Vol. 49, No. 2, Article 31, pp. 1-50, June 2017.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority oversampling technique," *J. Artif. Int. Res.*, Vol. 16, No. 1, pp. 321-357, Jan. 2002.

[5] D. Chicco, G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, "*BMC Genomics*, Vol. 21, No. 6, pp. 1-13, 2020.

[6] W. W. Cohen,"Fast effective rule induction," *Twelfth International Conference on Machine Learning*, pp. 115-123,1995.

[7] A. Cano, A. Zafra, and S. Ventura, "An interpretable classification rule mining algorithm," *Information Sciences*, Vol. 240, pp. 1-20, Aug. 2013.

[8] M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *Journal of Machine Learning Research*, Vol. 15, pp. 3133-3181, Oct. 2014.

[9] M. A. Hall, and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437-1447, Nov.-Dec. 2003.

[10] P. L. Hammer and T. O. Bonates, "Logical analysis of data– An overview: From combinatorial optimization to medical applications," *Annals of Operations Research*, Vol. 148, No. 1. pp. 203-225, Nov. 2006.

[11] S. J. Hong, R. G. Cain, and D. L. Ostapko, "MINI: A heuristic approach for logic minimization," *IBM J. Res. and Develop.*, pp. 443-458, Sept. 1974.

[12] S. J. Hong,"R-MINI: An iterative approach for generating minimal rules from examples," *IEEE Trans. Knowl. Data Eng.* Vol. 9, No. 5. pp. 709-717, 1997.

[13] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*,Vo. 5, pp. 221-232, 2016.

[14] X. Liu and Z. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," *Sixth International Conference on Data Mining* (ICDM'06), pp. 970-974, 2006.

[15] J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann Publishers, San Mateo, California, 1993.

[16] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[17] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[18] T. Sasao, "On a minimization of variables to represent sparse multi-valued input decision functions," *International Symposium on Multiple-Valued Logic ISMVL-2019*, Fredericton, Canada, pp. 182-187, May 21-23, 2019.

[19] T. Sasao, *Index Generation Functions*, Morgan & Claypool, Oct. 2019.

[20] T. Sasao, "Easily reconstructable logic functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2023), May 2023.

[21] T. Sasao, A. Holmgren, and P. Eklund, "A logical method to predict outcomes after coronary artery bypass grafting," *International Symposium on Multiple-Valued Logic*, May 2023.

[22] https://archive.ics.uci.edu/ml/datasets.php (Accessed Feburary 2, 2023)

[23] https://www.cs.waikato.ac.nz/ml/weka/index.html (Accessed Feburary 2, 2023)

[24] X. Wu, V. Kumar, and J R. Quinlan, et al.,"Top 10 algorithms in data mining," *Knowledge and Information Systems*, Vol. 14, pp. 1-37, 2008.