A Method To Generate Rules From Examples

Tsutomu Sasao
Department of Computer Science
Meiji University, Kawasaki, 214-8571, Japan

***Abstract*—Consider a set of distinct vectors of numbers, where each vector has a corresponding class. Such a set of vectors shows the sample data of a classification function. We developed a system that produces multiple-valued logical expressions from a set of examples. First, the set of vectors with real numbers is converted into a set of vectors with integers. Then, each class is represented by a simplified logical expression. Experimental results using UCI benchmark functions are shown. This system produces simpler rules than decision tree-based methods.**

***Index Terms*—data mining, logic minimization, multi-valued logic,discretization, domain reduction, incompletely specified function, classification,**

I. Introduction

Given a set of data, data mining is a technique to find a set of useful rules to represent the data.

Example 1.1: Table 1.1 shows the results of blood test for 10 healthy people and 10 people affected with liver cirrhosis. Three tests are used to detect liver cirrhosis. ZTT (Zinc sulfate Turbidity Test) indicates chronic hepatitis or liver cirrhosis when the value is greater than 12.0. ALT (Alanine Aminotransferease Test) indicates liver damage. ALT value greater than 30 indicates some problem. ALB (Albumin blood test) measures the amount of albumin. ALB value lower than 4.0 indicates liver disease or infection. In the diagnosis column of Table 1.1, 1 shows normal, while 2 shows liver cirrhosis.

To derive simple rules to detect liver cirrhosis, decision trees are often used. First, ZTT is used to partition the people into two classes. If ZTT is greater than 12.15, then the person is affected with liver cirrhosis. (Strictly speaking, the person with ID 12 is normal, and other 8 peoples are affected with liver cirrhosis.) If ZTT is less or equal to 12.15, then ALB is used to partition the people into two classes. If ALB is greater than 3.75, then the person is normal. Otherwise, the person is affected with liver cirrhosis. Figure 1.1 shows the decision tree to find liver cirrhosis. ■

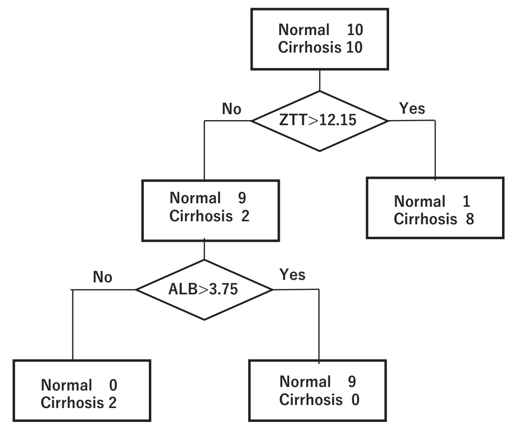

Fig. 1.1. Decision tree to find liver cirrhosis [17].

TABLE 1.1
RESULT OF BLOOD TEST [17]

ID	ZTT	ALT	ALB	Diagnosis
1	10.6	25	4.9	1
2	11.2	33	4.9	1
3	11.5	18	4.0	1
4	11.6	22	5.5	1
5	11.6	25	4.4	1
6	11.7	28	4.4	1
7	11.7	37	4.7	1
8	11.7	30	3.7	2
9	11.9	30	4.8	1
10	11.9	35	3.6	2
11	12.1	30	3.8	1
12	12.2	32	4.3	1
13	12.2	34	4.1	2
14	12.2	35	4.4	2
15	12.4	23	3.5	2
16	12.5	37	3.5	2
17	12.6	32	3.3	2
18	12.8	41	3.9	2
19	12.9	28	3.7	2
20	13.3	36	4.1	2

Diagnosis: 1: Normal, 2: Liver Cirrhosis.

C4.5 [9] and CART (Classification and regression tree) [2] are algorithms to derive decision trees from the set of integer vectors[1]. C4.5 uses entropy to find the decision variables, while CART uses Gini index. Rules can be derived from the decision trees.

This paper shows an alternative method to derive such rules. The method consists of three steps.

1) **Discretization**. Convert the data consisting of real numbers into that of integers (Table 3.1).
2) **Domain reduction**. Merge the intervals to reduce the dynamic range of the variables (Table 3.2).
3) **Multi-valued logic minimization**. Simplify the table, and derive expressions for each class, using techniques of logic minimization for partially defined functions.

Related research can be found in [4], [6], [7], [12] and [16]. The rest of this paper is organized as follows: Section II introduces words used in this paper. Section III shows the algorithms used in this method. Examples are also shown. Section IV shows the experimental results. Section V concludes the paper.

[1]Both C4.5 and CART works on only the integer data. To make the decision tree for Table 1.1, we must discretize the data by a separate algorithm.

II. Definitions

We solve problems in data mining using techniques of logic synthesis. Thus, the same notion is often called differently.

A. Logic Synthesis

In this part, we review words used in logic synthesis.

Definition 2.1: A multiple-valued input classification function is a mapping $f : \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_n \rightarrow \mathcal{M}$ where $\mathcal{P}_i = \{1, \ldots, p_i\}$ and $\mathcal{M} = \{1, 2, \ldots, m\}$.

Example 2.1: An automobile dealer has various models of a car. Each model is classified by the features shown in Table 2.1. Among these models, the following 4 models are in the inventory: (manual, 2 doors, white), (manual, 3 doors, blue), (manual, 3 doors, black), (manual, 4 doors, red). The following 5 models are available, but not in the inventory: (automatic, 2 doors, white), (automatic, 2 doors, black), (automatic, 3 doors, blue), (automatic, 3 doors, black), (automatic, 4 doors, red). In this table, X_1 shows the transmission type, X_2 shows the number of doors, and X_3 shows the color. If the car model is in the inventory, then $F = 1$, available but not in the inventory, then $F = 2$, otherwise (not available) $F = 3$, where X_1 is two-valued, X_2 is three-valued, and X_3 is four-valued. ■

TABLE 2.1
FEATURES OF AUTOMOBILES.

		1	2	3	4
X_1	Transmission	Manual	Automatic		
X_2	# of doors	2 doors	3 doors	4 doors	
X_3	Color	White	Blue	Red	Black

Definition 2.2: Let X be a variable that takes one value in $\mathcal{P} = \{1, 2, \ldots, p\}$. Let S be a subset ($S \subseteq P$) of P. Then, X^S is a **literal** of X. When $X \in S$, $X^S = 1$, and when $X \notin S$, $X^S = 0$. Let $S_i \subseteq \mathcal{P}_i$ $(i = 1, 2, \ldots, n)$, then ${X_1}^{S_1}{X_2}^{S_2} \cdots {X_n}^{S_n}$ is a **product**. $\bigvee_{(S_1, S_2, \ldots, S_n)} {X_1}^{S_1}{X_2}^{S_2} \cdots {X_n}^{S_n}$ is a **sum-of-products expression** (**SOP**). When $S_i = \mathcal{P}_i$, ${X_i}^{S_i} = 1$ and the product is independent of X_i. In this case, literal ${X_i}^{P_i}$ is redundant and can be deleted. A product corresponds to a **cube**. When $|S_i| = 1$ $(i = 1, 2, \ldots, n)$, a product corresponds to an element of the domain. This product is a **minterm of** f. When $S_i = \mathcal{P}_i$ $(i = 1, 2, \ldots, n)$, the product corresponds to the constant 1. This product corresponds to a **universal cube**. **Cube size** is the total number of vertices contained in the cube. When $p_i = 2$ $(i = 1, 2, \ldots, n)$, a function is a two-valued logic function. An arbitrary multi-valued input classification function is represented by an SOP. Many SOPs exist that represent the same function. Among them, the one with the minimum number of products is the **minimum SOP** (**MSOP**).

Example 2.2: Consider the automobile dealer in the previous example. Let $\mathcal{P}_1 = \{1, 2\}$, $\mathcal{P}_2 = \{1, 2, 3\}$, $\mathcal{P}_3 = \{1, 2, 3, 4\}$. Let F_1 be the set of automobiles in the inventory. Let F_2 be the set of available automobiles, but not in the inventory. Let F_3 be the set of automobiles that are unavailable. Then, we have

$F_1 = \{(1, 1, 1), (1, 2, 2), (1, 2, 4), (1, 3, 3)\}$,
$F_2 = \{(2, 1, 1), (2, 1, 4), (2, 2, 2), (2, 2, 4), (2, 3, 3)\}$, and
$F_3 = \{\text{Other combinations}\}$.

The size of the universe is

$$p_1 \times p_2 \times p_3 = 2 \times 3 \times 4 = 24.$$

■

Example 2.3: Consider the set of automobiles in the previous example. It represents a multi-valued input classification function,

$$\begin{aligned} F_1 &= {X_1}^{\{1\}}{X_2}^{\{1\}}{X_3}^{\{1\}} \vee {X_1}^{\{1\}}{X_2}^{\{2\}}{X_3}^{\{2\}} \vee \\ &\quad {X_1}^{\{1\}}{X_2}^{\{2\}}{X_3}^{\{4\}} \vee {X_1}^{\{1\}}{X_2}^{\{3\}}{X_3}^{\{3\}} \\ &= {X_1}^{\{1\}}{X_2}^{\{1\}}{X_3}^{\{1\}} \vee {X_1}^{\{1\}}{X_2}^{\{2\}}{X_3}^{\{2,4\}} \vee \\ &\quad {X_1}^{\{1\}}{X_2}^{\{3\}}{X_3}^{\{3\}} \end{aligned}$$

and

$$\begin{aligned} F_2 &= {X_1}^{\{2\}}{X_2}^{\{1\}}{X_3}^{\{1\}} \vee {X_1}^{\{2\}}{X_2}^{\{1\}}{X_3}^{\{4\}} \vee \\ &\quad {X_1}^{\{2\}}{X_2}^{\{2\}}{X_3}^{\{2\}} \vee {X_1}^{\{2\}}{X_2}^{\{2\}}{X_3}^{\{4\}} \vee \\ &\quad {X_1}^{\{2\}}{X_2}^{\{3\}}{X_3}^{\{3\}} \\ &= {X_1}^{\{2\}}{X_2}^{\{1\}}{X_3}^{\{1,4\}} \vee {X_1}^{\{2\}}{X_2}^{\{2\}}{X_3}^{\{2,4\}} \vee \\ &\quad {X_1}^{\{2\}}{X_2}^{\{3\}}{X_3}^{\{3\}}. \end{aligned}$$

■

Let the domain of a multi-valued input two-valued output function be $\mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_n$. In this case, the product $c = X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n}$, where $S_i \subseteq P_i$ corresponds to a **cube** in an n-dimensional hyper-cube. The **bit representation** (positional cube notation) of a cube c is the concatenation of binary numbers showing the cube. $c = \pi_1$-π_2- $\cdots$ -π_n, where $\pi_i = (\xi_0 \xi_1 \cdots \xi_{p_i - 1})$, such that

$$\xi_j = \begin{cases} 1 & (\text{when } j \in S_i), \\ 0 & (\text{when } j \notin S_i). \end{cases}$$

The bit representation of an SOP is an **array**. An array is set of cubes.

Example 2.4: The last SOP for F_2 of the previous example is represented by an array:

$$\begin{array}{c} X_1 \quad X_2 \quad X_3 \\ 12 \text{ - } 123 \text{ - } 1234 \\ \begin{bmatrix} 01 \text{ - } 100 \text{ - } 1001 \\ 01 \text{ - } 010 \text{ - } 0101 \\ 01 \text{ - } 001 \text{ - } 0010 \end{bmatrix} \end{array}$$

Note that a variable with all 1's in the bit representation can be omitted in the SOP. ■

Definition 2.3: An SOP is called a disjoint sum-of-products expression (DSOP), if all the products are mutually disjoint.

Theorem 2.1 ([10]): To represent an $(n = 2r)$-variable logic function

$$x_1 x_2 \vee x_3 x_4 \vee \cdots \vee x_{n-1} x_n,$$

an SOP requires r products, while a DSOP requires $2^r - 1$ products.

B. Data Mining

In this part, we review terminology used in data mining.

Table 2.2 shows the relations of words used in three different specializations.

The subject of this paper is stated as follows: Given a set of examples (minterms), derive a minimal set of rules (products) that covers examples. We are interested in deriving simple rules. Simpler rules are more understandable and more efficient to apply [6].

Definition 2.4: 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 2.5: A **categorical** variable is a variable that can take one of limited number of possible values. A **continuous** variable is a variable that may take on any value within a finite or infinite interval. **Discretization** [8] is to convert numerical variables into categorical ones.

Example 2.5: The type of transmission and colors in Example 2.1 are categorical variables. Length, weights, temperature, and time are continuous variables. ZTT, ALT, and ALB in Example 1.1 are continues variables. ■

Definition 2.6: Two examples are **inconsistent** if they have the same input parts, but belong to different classes.

Example 2.6: Given a decision tree, the set of rules can be derived from the path from the root node to a leaf. Fig. 1.1 produces three rules:

Rule 1: If $(ZTT \leq 12.15)$ and $(ALB \leq 3.75)$, then Cirrhosis.
Rule 2: If $(ZTT \leq 12.15)$ and $(ALB > 3.75)$, then Normal.
Rule 3: If $(ZTT > 12.15)$, then Cirrhosis.

This set of rules is complete, but produces a wrong result for the patient with $ID = 12$.

Reduction of the number of rules corresponds to logic minimization. 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. The numbers of examples in each classes range in the thousands to millions. The size of the universe in data mining can be larger than that of a typical logic design. For example, *Dermatology* that appears in Section IV has 34 variables. After the domain reduction, 21 variables take 4 values, 8 variables take 3 values, 4 variables take 2 values, and one variable takes 59 values. Thus, the size of the universe is

$$4^{21} \times 3^{12} \times 2^4 \times 59 \simeq 2.723 \times 10^{19}.$$

This is approximately equal to a binary logic function with $n = 65$ variables, since

$$2^{65} \simeq 3.6 \times 10^{19}.$$

TABLE 2.2
TERMINOLOGY IN DIFFERENT AREAS OF SPECIALIZATION.

Logic design	Geometry	Data mining
minterm	vertex	example, instance, sample, object
implicant	cube	rule
prime implicant	prime cube	maxunakkt general rule
SOP	covering cubes	covering rule set
variable	variable	feature, attribute

III. Algorithms and Examples

A. Discretization

Algorithm 3.1: (Discretization)

1) Let $\vec{R}(i) = (R_1(i), R_2(i), \ldots, R_n(i))$ be the data for the i-th example, where $i = 1, 2, \ldots, k$, and let $Class(i)$ be the class of the i-th example.
2) For $j = 1$ to n do the followings:
3) Sort the values of $\vec{R}_j(i), (i = 1, \ldots, k)$ in ascending order.
4) Let $N(j)$ be the number of distinct elements in $\vec{R}_j(i), (i = 1, \ldots, k)$.
5) Let $\vec{X}(i) = (X_1(i), X_2(i), \ldots, X_n(i))$ be the discretized data for $\vec{R}(i)$. Assign natural numbersto $X_j(i)$ as follows: To the smallest $R_j(i)$ assign 1. To the second smallest $R_j(i)$ assign 2. To the largest $R_j(i)$ assign $N(j)$.

Example 3.1: Consider Table 1.1. In this case, ZTT, ALT, and ALB correspond to R_1, R_2 and R_3, respectively. Note that the examples are already sorted with respect to the value of ZTT. So, for the first example, $X_1(1) = 1$. Similarly, $X_1(2) = 2$ and $X_1(3) = 3$. Since $R_1(4) = R_1(5)$, we have $X_1(4) = X_1(5) = 4$. Also, $R_1(6) = R_1(7) = R_1(8)$, we have $X_1(6) = X_1(7) = X_1(8) = 5$, and so on. Note that $X_1(20) = 14$, since $N(1) = 14$. In a similar way, X_2 and X_3 are derived. In this way, we have Table 3.1. ■

TABLE 3.1
BLOOD TEST AFTER DISCRETIZATION.

ID	X_1	X_2	X_3	Diagnosis
1	1	4	13	1
2	2	8	13	1
3	3	1	7	1
4	4	2	14	1
5	4	4	10	1
6	5	5	10	1
7	5	12	11	1
8	5	6	4	2
9	6	6	12	1
10	6	10	3	2
11	7	6	5	1
12	8	7	9	1
13	8	9	8	2
14	8	10	10	2
15	9	3	2	2
16	10	12	2	2
17	11	7	1	2
18	12	13	6	2
19	13	5	4	2
20	14	11	8	2

Diagnosis: 1: Normal, 2: Liver Cirrhosis.

B. Domain Reduction

Algorithm 3.2: (Domain Reduction)

1) Let $\vec{X}(i) = (X_1(i), X_2(i), \ldots, X_n(i))$ be the data for the i-th example, where $i = 1, 2, \ldots, k$, and let $Class(i)$ be the class of the i-th example.
2) For $j = 1$ to n do the followings:
3) Sort the values of $\vec{X}_j(i), (i = 1, 2, \ldots, k)$ in ascending order.
4) Let $\vec{Y}(i) = (Y_1(i), Y_2(i), \ldots, Y_n(i))$ be the domain-reduced data for $\vec{X}(i)$. $Y_j(i)$ is formed as follows: Let $X_j(1)$ be the smallest value, assign $Y_j(1)$ to 1. If $X_j(i+r) = X_j(i)+r$ and $Class(i+r) = Class(i)$, then $Y_j(i+r) \leftarrow Y_j(i)$. Otherwise, $Y_j(i+r) \leftarrow Y_j(i) + 1$.

Example 3.2: Consider Table 3.1. Since $Class(i) = 1$ for $i = 1, 2, 3, 4, 5$, we have $Y_1(i) = 1$ for $i = 1, 2, 3, 4, 5$. However, $X_i(6) = X_i(7) = X_i(8)$, but $Class(6) \neq Class(8)$, so $Y_1(6) = 2$. Also $X_1(9) = X_1(10)$, but $Class(9) \neq Class(10)$, so $Y_1(9) = 3$. In this way, we have Table 3.2. Note that the maximum values for Y_1, Y_2, and Y_3 are reduced to 6, 10, and 8, respectively. ■

TABLE 3.2
BLOOD TEST AFTER DOMAIN REDUCTION.

ID	Y_1	Y_2	Y_3	Diagnosis
1	1	3	8	1
2	1	7	8	1
3	1	1	4	1
4	1	1	8	1
5	1	3	7	1
6	2	4	7	1
7	2	9	8	1
8	2	5	1	2
9	3	5	8	1
10	3	8	1	2
11	4	5	2	1
12	5	6	6	1
13	5	8	5	2
14	5	8	7	2
15	6	2	1	2
16	6	9	1	2
17	6	6	1	2
18	6	10	3	2
19	6	4	1	2
20	6	8	5	2

Diagnosis: 1: Normal, 2: Liver Cirrhosis.

C. Multi-Valued Logic Minimization

Algorithm 3.3: (Simplification of Multi-valued input Classification Function)

1) Let m be the number of classes.
2) Partition the function into $F_1, F_2, \ldots, F_m$.
3) For each function F_i, simplify the expression by MINI5 algorithm[2]. In this case, F_i is treated as the ON set, while $\cup_{j=1, j \neq i}^{m} F_j$ is treated as the OFF set.
4) Expand the input parts.
5) Merge the functions F_i $(i = 1, 2, \ldots, m)$ to form a multi-class function.

Example 3.3: Consider the function shown in Table 3.2. In this case, Y_1 takes 6 values, Y_2 takes 10 values, and Y_3 takes 8 values. So, the size of the universe, i.e., the total number of input combinations, is $6 \times 10 \times 8 = 480$. However, F_1 and F_2 are specified by 10 combinations each. For the remaining $480 - 20 = 460$ combinations, the function values are undefined. Thus, the function is very sparse. The positional cube notation of the original data is shown below:

$$
\begin{array}{cccc}
Y_1 & Y_2 & Y_3 & \\
123456 & 1234567890 & 12345678 &
\end{array}
$$

$$
\left[\begin{array}{cccc}
100000 & 0010000000 & 00000001 & 10 \\
100000 & 0000001000 & 00000001 & 10 \\
100000 & 1000000000 & 00010000 & 10 \\
100000 & 1000000000 & 00000001 & 10 \\
100000 & 0010000000 & 00000010 & 10 \\
010000 & 0001000000 & 00000010 & 10 \\
010000 & 0000000010 & 00000001 & 10 \\
010000 & 0000100000 & 10000000 & 01 \\
001000 & 0000100000 & 00000001 & 10 \\
001000 & 0000000100 & 10000000 & 01 \\
000100 & 0000100000 & 01000000 & 10 \\
000010 & 0000010000 & 00000100 & 10 \\
000010 & 0000000100 & 00001000 & 01 \\
000010 & 0000000100 & 00000010 & 01 \\
000001 & 0100000000 & 10000000 & 01 \\
000001 & 0000000010 & 10000000 & 01 \\
000001 & 0000010000 & 10000000 & 01 \\
000001 & 0000000001 & 00100000 & 01 \\
000001 & 0001000000 & 10000000 & 01 \\
000001 & 0000000100 & 00001000 & 01
\end{array}\right]
$$

After logic minimization, we have the following array:

$$
\begin{array}{cccc}
Y_1 & Y_2 & Y_3 & \\
123456 & 1234567890 & 12345678 &
\end{array}
$$

$$
\left[\begin{array}{cccc}
111111 & 1111111011 & 01011111 & 10 \\
111111 & 1111111111 & 10101000 & 01 \\
111111 & 1100111111 & 10101010 & 01
\end{array}\right]
$$

This array shows the expressions:

$$
\begin{aligned}
F_1 &= \overline{Y_2^{\{8\}}} \cdot \overline{Y_3^{\{1,3\}}} \\
F_2 &= Y_3^{\{1,3,5\}} \vee \overline{Y_2^{\{3,4\}}} \cdot Y_3^{\{1,3,5,7\}} \\
&= Y_3^{\{1,3,5\}} \vee \overline{Y_2^{\{3,4\}}} \cdot Y_3^{\{7\}}
\end{aligned}
$$

From these, we have the following rules:
Class: Normal: (Number of examples: 10)
Rule 1 ALT : NOT {34, 35, 36}
ALB : {3.8, or greater than 4.0}
$\Longrightarrow$ Normal (Coverage: 10)
Class: Liver Cirrhosis: (Number of examples: 10)
Rule 2 ALB :{3.3, 3.5, 3.6, 3.7, 3.9, 4.1}
$\Longrightarrow$ Cirrhosis (Coverage: 9)
Rule 3 ALT : NOT {25, 28}
ALB : {4.4}
$\Longrightarrow$ Cirrhosis (Coverage: 1)
These rules are consistent, and covers all the examples in Table 1.1. ■

IV. EXPERIMENTAL RESULTS

We developed programs to perform the procedures presented in the previous section, and applied to the UCI data set [18]. Table 4.1 shows experimental results. The first column

[2]MINI5 is a logic minimizer for machine learning. It is similar to MINI [5], but it uses the ON and the OFF sets as inputs. It treats variables with up to 480 values.

shows the name of the function; the second column shows n, the original numbers of variables; the third column shows k, the number of the instances; the fourth column shows m, the number of the classes; the fifth column shows n_1, the number of the variables after SOP minimization; the sixth column shows p_1, the number of the rules after SOP minimization; the seventh column shows n_2, the number of the variables after variable minimization [15]; and the last column shows p_2, the number of the rules after SOP minimization. Functions with * marks show that the numbers of variables were reduced by the algorithm [15], but the number of products increased.

The presented method is applicable to only consistent data sets. If there is a pair of inconsistent (conflicting) samples, one of them must be removed. The most time-consuming part of the procedure is the MVSOP minimization. After reducing the number of variables by [14], the minimization time for MVSOPs became shorter.

Alcohol data set contains five different types of alcohols. They are classified by QCM (quartz crystal microbalance) gas sensor. In this experiment, QCM3 was used to classify the data into five classes. Five rules were generated.

Bupa (liver) contains data for liver disorders. The first five variables are all blood tests which are sensitive to liver disorders that might arise from excessive alcohol consumption. The sixth variable shows the number of the half-pint equivalents of alcoholic beverages drunk per day

Caesarian contains information about caesarian section results of 80 pregnant women with the most important characteristics of delivery problems in the medical field. Four conflicting samples were removed from the original data. 20 rules were generated.

Dermatology data set contains 34 variables. The data is classified into six classes. 10 conflicting samples were removed from the original data. 10 rules were generated.

Ecoli contains data for protein localization sites. It has six classes.

Fertility data set contains semen samples provided by 100 volunteers. The samples were analyzed according to the WHO 2010 criteria. One conflicting sample was removed from the original data. 9 rules were generated.

Glass data set contains 214 samples for 6 different applications. This can be used in criminological investigation.

Inoshpere contains data for a radar system. This system consists of a phased array of 16 high-frequency antennas. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.

Iris data set contains three classes of 50 instances each, where each class refers to a type of iris plant [3]. One class is linearly separable from the other two; the latter are NOT linearly separable from each other. The variables are

R_1 : sepal length,
R_2 : sepal width,
R_3 : petal length, and
R_4 : petal width.

Three classes are

1) Iris Setosa,
2) Iris Versicolour, and
3) Iris Virginica.

8 rules were generated.

Fig. 4.1 shows the minimized array. The array shows the following expressions:

$$
\begin{aligned}
F_1 &= Y_4^{\{1\}} \\
F_2 &= \overline{Y_3^{\{9\}}} \cdot Y_4^{\{2,3,5\}} \vee \overline{Y_1^{\{2\}}} \cdot \overline{Y_3^{\{7,8\}}} \cdot Y_4^{\{2,4,6\}} \vee \\
&\quad \overline{Y_3^{\{3\}}} \cdot Y_4^{\{2,6\}} \vee \overline{Y_2^{\{8,10\}}} \cdot \overline{Y_3^{\{6,9\}}} Y_4^{\{2,3,5,7\}} \\
F_3 &= \overline{Y_2^{\{12,14\}}} \cdot \overline{Y_3^{\{4,8\}}} \cdot Y_4^{\{5,7,8\}} \vee \\
&\quad Y_3^{\{1,7,8,9\}} \cdot Y_4^{\{2,3,4,7,8\}} \vee \overline{Y_3^{\{7\}}} Y_4^{\{6,8\}}
\end{aligned}
$$

From these, we have the following rules:

Class: Iris Setosa: (Number of examples: 50)

Rule 1 Petal width: $\{0.1 \sim 0.6\}$
$\Longrightarrow$ Iris Setosa (Coverage: 50)

Class: Iris Versicolour : (Number of examples: 50)

Rule 2 Petal length: Less than 5.2
Petal width: $\{1.0 \sim 1.4, 1.6\}$
$\Longrightarrow$ Iris Versicolour (Coverage: 38)

Rule 3 Sepal length: NOT $\{4.9\}$
Petal length: NOT $\{5.0, 5.1\}$
Petal width: $\{1.0 \sim 1.3, 1.5, 1.7\}$
$\Longrightarrow$ Iris Versicolour (Coverage: 37)

Rule 4 Petal length: NOT $\{4.5\}$
Petal width: $\{1.0 \sim 1.3, 1.7\}$
$\Longrightarrow$ Iris Versicolour (Coverage: 28)

Rule 5 Sepal width: NOT $\{2.8, 3.0\}$
Petal length: NOT $\{2.6, 2.8\}$
Petal width: $\{1.0 \sim 1.4, 1.6, 1.7\}$
$\Longrightarrow$ Iris Versicolour (Coverage: 30)

Class: Iris Virginica : (Number of examples: 50)

Rule 6 Sepal width: NOT $\{3.2, 3.4\}$
Petal length: NOT $\{4.6, 4.7, 5.1\}$
Petal width: $\{1.6, 1.8 \sim 2.5\}$
$\Longrightarrow$ Iris Virginica (Coverage: 34)

Rule 7 Petal length: $\{1.0 \sim 1.9, 5.0 \sim 6.9\}$
Petal width: $\{1.0 \sim 1.5, 1.8 \sim 2.5\}$
$\Longrightarrow$ Iris Virginica (Coverage: 43)

Rule 8 Petal length: NOT $\{5.0\}$
Petal width: $\{1.7, 1.9 \sim 2.5\}$
$\Longrightarrow$ Iris Virginica (Coverage: 33)

These rules are consistent, and covers all the 150 examples in the data set. Note that many examples in Iris Versicolour and Iris Virginica are covered by multiple rules.

Shuttle (Satlog) contains data for five classes of shuttles.

Thyroid contains data for hypothyroid disease. There are three classes. Even after reducing the domain, one of the variables takes 237 values.

Wine set is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars.

$$
\begin{array}{ccccc}
Y_1 & Y_2 & Y_3 & Y_4 & \\
123456789012345678901234 & 1234567890123456789 & 123456789 & 12345678 & \\
\end{array}
$$
$$
\left[
\begin{array}{ccccc}
111111111111111111111111 & 1111111111111111111 & 111111111 & 10000000 & 100 \\
111111111111111111111111 & 1111111111111111111 & 111111110 & 01101000 & 010 \\
101111111111111111111111 & 1111111111111111111 & 111111001 & 01010100 & 010 \\
111111111111111111111111 & 1111111111111111111 & 110111111 & 01000100 & 010 \\
111111111111111111111111 & 1111110101111111111 & 111110110 & 01101010 & 010 \\
111111111111111111111111 & 1111111111101011111 & 111011101 & 00001011 & 001 \\
111111111111111111111111 & 1111111111111111111 & 100000111 & 01110011 & 001 \\
111111111111111111111111 & 1111111111111111111 & 111111011 & 00000101 & 001
\end{array}
\right]
$$

Fig. 4.1. Simplified expressions for iris data set.

TABLE 4.1
EXPERIMENTAL RESULTS.

				SOP Min		Var Min SOP Min		
	n	k	m	n_1	p_1	n_2	p_2	
Alcohol	11	25	5	1	5	1	5	
Bupa (liver)	6	341	2	6	26	3	37	$*$
Caesarian	5	76	2	5	20	5	20	
Dermatology	34	356	6	17	10	6	60	$*$
Ecoli	7	332	6	6	31	3	48	$*$
Fertility	9	99	2	4	9	4	9	
Glass	9	214	6	6	14	2	37	$*$
Inosphere	33	351	2	13	2	2	3	$*$
Iris	4	150	3	4	8	3	10	$*$
Shuttle	9	43500	5	9	11	4	18	$*$
Thyroid	21	7200	3	7	18	3	39	$*$
Wine	13	178	3	4	3	2	12	$*$

V. CONCLUSIONS AND COMMENTS

This paper showed a method to derive rules for a given set of examples. Unlike conventional methods that use decision trees, it first reduces the domain, and then produces a sparsely defined discrete function. Then, multiple-valued input expressions are simplified. The method produces consistent and a complete set of rules for a given consistent set of examples. Thus, the rules produce correct results for all the examples.

For many functions, we could reduce the numbers of variables before SOP minimization [15]. With this, CPU time for SOP minimization was reduced drastically. However, the reduction of variables increased the number of the products. The CPU time for multi-valued SOP minimization is approximately proportional to $n_i p_i^2$, where n_i is the number of variables after minimization, and p_i is the number of products after minimization.

Note that the tree-based method produces a disjoint sum-of-products expressions (DSOPs) [10], while the present method produces sum-of-products expressions (SOPs). Thus, the presented method tends to produce fewer rules than the tree-based method. The presented method is useful for analyzing properties of data in various fields.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. Reviewer's comments were useful for the improvement of the paper.

REFERENCES

[1] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik, "An implementation of logical analysis of data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 2, March 2000, pp. 292-306.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, CRC Press 1984.

[3] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics*, 7, Part II, 179-188 (1936).

[4] 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.

[5] 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.

[6] 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.

[7] S. Kahramanli, M. Hacibeyoglu, and A. Arslan, "A Boolean function approach to feature selection in consistent decision information systems," *Expert Systems with Applications: An International Journal*, " Vol. 38, Issue 7, July, 2011, pp. 8229-8239.

[8] H. Liu, F, Hussain, C. L. Tan and M. Dash, "Discretization: An Enabling Technique," *Data Mining and Knowledge Discovery*, Vol. 6, pp. 393-423, 2002.

[9] J. R. Quinlan, *C4.5: Program for Machine Learning*, San Mateo, Morgan Kaufmann 1993.

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

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

[12] 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, May 21-23, 2019, pp. 182-187.

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

[14] T. Sasao, "On the minimization of variables to represent partially defined classification functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2020), May 20-22, Miyazaki, Japan.

[15] T. Sasao, "Reduction methods of variables for large-scale classification functions," in *International Workshop on Logic and Synthesis*, July 2020.

[16] E. Triantaphyllou, *Data Mining and Knowledge Discovery via Logic-Based Methods:* Theory, Algorithms, and Applications, Springer, 2010.

[17] O. Uchida, *Introduction to Data Mining*, (in Japanese), Nihon Keizai Shinbun, 2002.

[18] https://archive.ics.uci.edu/ml/datasets.php