On a Minimization of Variables to Represent Sparse Multi-Valued Input Decision Functions

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

***Abstract*—A multiple-valued input decision function is a mapping $f : P^n \rightarrow \{0,1\}$, where $P = \{0,1,\ldots,p-1\}$. This paper considers the learning of such a function. That is, given the TRUE-set $T \subseteq P^n$ and the FALSE-set $F \subseteq P^n$, obtain a function f such that $f(\vec{a}) = 1$ for any $\vec{a} \in T$, and $f(\vec{b}) = 0$ for any $\vec{b} \in F$. We show a method to find a function such that f depends on the least number of variables. Applications of such functions include detection of poisonous mushrooms, hepatitis and breast cancer.**

***Index Terms*—data mining, logic minimization, machine learning, monotone increasing function, multi-valued logic, partially defined function, support minimization.**

I. Introduction

Given two disjoint sample sets, we seek a simple rule to distinguish them. Such a work is the main goal of machine learning and data mining. Occam's razor argument [2] uses a simple rule: Given two explanations of the data, all other things being equal, the simpler explanation is preferable [1]. Simpler rules are more understandable and more efficient to apply.

As for the complexity of the rule, various measures exist: The number of the nodes in a decision tree [12], [20], [21]; the number of the products in a sum-of-products expression (SOP) or a disjunctive normal form (DNF) [7]; the number of the clauses in a product-of-sums expression (POS) or a conjunctive normal form (CNF) [27]; the number of the variables to represent the rule, etc.

In this paper, we minimize the number of variables to represent the rule. The rest of the paper is organized as follow: Section II discusses definitions used; Section III shows a method to reduce the number of variables to represent the function; Section IV shows a method to make a consistent training set for monotone increasing functions; Section V shows an application to detect poisonous mushrooms; Section VI shows an application to determine the prognosis of someone who has hepatitis; Section VII shows an application to detect breast cancer; and Section VIII concludes the paper.

Terminology of this paper are based on [13], [22], [25], [27]. Note that the terminology of data science is different from that of logic design. For people working in logic design, the following translations are useful: A partially defined function corresponds to an incompletely specified function. Training data corresponds to a specification of a design. Removal of inconsistency does not exist in logic design, since in logic design the specifications are assumed to be consistent and correct.

II. Definitions

Definition 2.1: A multiple-valued input decision function is a mapping $f : \mathcal{P}^n \rightarrow \mathcal{B}$ where $\mathcal{P} = \{0,1,\ldots,p-1\}$ and $\mathcal{B} = \{0,1\}$.

Definition 2.2: A pair of subsets (T,F) such that $T \subseteq \mathcal{P}^n$, $F \subseteq \mathcal{P}^n$, and $T \cap F = \phi$ is **training data**. If $T \cup F \subset \mathcal{P}^n$, then the pair is a **partially defined function**. If $T \cup F = \mathcal{P}^n$, then the pair is a **totally defined function**.

Definition 2.3: For a partially defined function (T,F), a function f satisfying

$$T(f) \supseteq T \text{ and } F(f) \supseteq F$$

is an **extension** of (T,F), where

$$\begin{array}{lcl} T(f) & = & \{\vec{a} \in \mathcal{P}^n | f(\vec{a}) = 1\}, \quad \text{and} \\ F(f) & = & \{\vec{b} \in \mathcal{P}^n | f(\vec{b}) = 0\}. \end{array}$$

Learning of a function is to find an extension f of (T,F).

Example 2.1: Let

$$\begin{array}{l} T = \{(2,2,0,1),(2,1,1,2)\}, \quad \text{and} \\ F = \{(1,1,0,1),(0,1,1,2)\}, \end{array}$$

be training data. Then, the function f, where

$$\begin{array}{ll} T(f) = \{(2,*,*,*)\}, & \text{and} \\ F(f) = \{(1,*,*,*),(0,*,*,*)\} & \end{array}$$

is an extension of (T,F). Note that * denotes 0, 1 or 2. ■

Definition 2.4: [22] f is **monotone increasing** if $f(\vec{a}) \geq f(\vec{b})$ for any $\vec{a}$, $\vec{b} \in \mathcal{P}^n$ such that $\vec{a} \geq \vec{b}$.

Definition 2.5: Let $\mathcal{P} = \{0,1,\ldots,p-1\}$. Let X be a variable that takes its value from $\mathcal{P} = \{0,1,\ldots,p-1\}$. Let $S \subseteq \mathcal{P}$, then, X^S is a **literal** of X. When $X \in S$, $X^S = 1$, and when $X \notin S$, $X^S = 0$. Let $S_i \subseteq \mathcal{P}$ for $i = 1,2,\ldots,n$, then ${X_1}^{S_1}{X_2}^{S_2}\cdots{X_n}^{S_n}$ is a **logical 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 $S = \mathcal{P}$, $X^S = 1$ and the literal is independent of X. In this case, the literal $X^{\mathcal{P}}$ is redundant and can be deleted. When $|S_i| = 1$ $(i = 1,2,\ldots,n)$, a logical product corresponds to an element of the domain. This product is a **minterm of** f. When $S_i = \mathcal{P}$ $(i = 1,2,\ldots,n)$, the logical product corresponds to the constant 1.

An arbitrary multi-valued input decision 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**).

TABLE 2.1
RESULT OF MEDICAL TEST.

	X_1	X_2	X_3
T	2	1	1
	1	2	1
	1	1	2
F	1	0	0
	0	1	0
	0	0	1

TABLE 2.2
VARIABLES SHOWING THE RESULTS OF ENTRANCE EXAMINATION.

Variable	Subject
X_1	Physics
X_2	Mathematics
X_3	English

Example 2.2: Consider T and F shown in Table 2.1. It represents a multi-valued input decision function, where $f : \mathcal{P}^3 \to \mathcal{B}, \mathcal{P} = \{0,1,2\}$ and $\mathcal{B} = \{0,1\}$. X_i takes three values, where $i = 1,2,3$. T can be represented by [22]

$$\begin{aligned} f_1 &= X_1{}^{\{2\}}X_2{}^{\{1\}}X_3{}^{\{1\}} \vee X_1{}^{\{1\}}X_2{}^{\{2\}}X_3{}^{\{1\}} \\ &\vee X_1{}^{\{1\}}X_2{}^{\{1\}}X_3{}^{\{2\}}, \end{aligned}$$

while F can be represented by

$$\begin{aligned} f_2 &= X_1{}^{\{1\}}X_2{}^{\{0\}}X_3{}^{\{0\}} \vee X_1{}^{\{0\}}X_2{}^{\{1\}}X_3{}^{\{0\}} \\ &\vee X_1{}^{\{0\}}X_2{}^{\{0\}}X_3{}^{\{1\}}. \end{aligned}$$

Monotone increasing extensions of (T,F) include

$$\begin{aligned} f_3 &= X_1{}^{\{2\}}X_2{}^{\{1,2\}}X_3{}^{\{1,2\}} \vee X_1{}^{\{1,2\}}X_2{}^{\{2\}}X_3{}^{\{1,2\}} \\ &\vee X_1{}^{\{1,2\}}X_2{}^{\{1,2\}}X_3{}^{\{2\}}, \\ f_4 &= X_1{}^{\{1,2\}}X_2{}^{\{1,2\}}X_3{}^{\{1,2\}}, \\ f_5 &= X_1{}^{\{1,2\}}X_2{}^{\{1,2\}} \vee X_2{}^{\{1,2\}}X_3{}^{\{1,2\}}, \\ &\vee X_1{}^{\{1,2\}}X_3{}^{\{1,2\}}, \text{ and} \\ f_6 &= X_1{}^{\{1,2\}}X_2{}^{\{1,2\}}. \end{aligned}$$

■

Example 2.3: In Table 2.1, assume that each component of a point $\vec{a}$ shows the result of one of three entrance examinations shown in Table 2.2. T shows the set of students who passed the examination, while F shows the set students who failed the examination. The larger the value of a component, the more likely the student passed the examination. Thus, the function can be considered as monotone increasing.

Prof. A recommends to use f_3 as the requirement to pass the examination. However, Prof. B recommends to use f_4 as the requirement to pass the examination. For Prof. B, the students with the result $(1,1,1)$ pass the examination. Prof. C recommends to use f_5 as the requirement to pass the examination. For Prof. C, the students with the scores $(0,1,1),(0,1,0)$ or $(1,1,0)$ also pass the examination. Prof. D recommends to use f_6 as the requirement to pass the examination. In this case, the result of English (X_3) is not used for the decision. ■

Definition 2.6: For a subset $U \subseteq \mathcal{P}^n$ and $S \subseteq \{1,2,\ldots,n\}$, we denote by $U|_S$ the **projection** of U to S. In other words, $U|_S = \{\vec{a}|_S\}$, where $\vec{a}|_S$ is the point obtained from $\vec{a} \in U$ by considering only those components a_j with $j \in S$.

TABLE 2.3
EXAMPLE FOR SUPPORT REDUCTION.

		X_1	X_2	X_3	X_4
T	$\vec{a}_1$	1	2	0	1
	$\vec{a}_2$	0	1	1	2
F	$\vec{b}_1$	1	1	0	1
	$\vec{b}_2$	0	1	2	2

Example 2.4: Let $\mathcal{P} = \{0,1,2\}$ and $n = 4$. Let $U = \{(1,2,0,1),(0,1,1,2),(2,0,1,2)\}$ and $S = \{2,3\}$. Then, we have $U|_S = \{(*,2,0,*),(*,1,1,*),(*,0,1,*)\}$. ■

We are interested in finding a function f depending on the least number of variables.

Definition 2.7: Given a partially defined function (T,F) with $T,F \subseteq \mathcal{P}^n$, a subset $S \subseteq \{1,2,\ldots n\}$ is a **support set**[1] if $T|_S \cap F|_S = \phi$. In such a case, all variables $x_j, j \in \overline{S}$ are **redundant** because $(T|_S, F|_S)$ does not depend on them.

Example 2.5: Let (T,F) be the function in Table 2.3. Then, $S = \{2,3\}$ is a support set, since $T|_S \cap, F|_S = \phi$, where

$$\begin{aligned} T|_S &= \{(*,2,0,*),(*,1,1,*)\} \quad \text{and} \\ F|_S &= \{(*,1,0,*),(*,1,2,*)\}. \end{aligned}$$

In this case, the function is represented by the expression:

$$f = X_2^{\{0,2\}} \vee X_2^{\{1\}}X_3^{\{1\}},$$

where

$$\begin{aligned} T(f) &= \{(*,0,*,*),(*,2,*,*),(*,1,1,*)\} \supseteq T|_S, \\ F(f) &= \{(*,1,0,*),(*,1,2,*)\} \supseteq F|_S, \end{aligned}$$

and * denotes either 0, 1, or 2. ■

In the next section, we show that $\{2,3\}$ is the minimum support set.

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

$$\begin{aligned} \vec{a} &= (a_1, a_2, \ldots, \quad a_i, \quad \ldots, a_n), \\ \vec{b} &= (b_1, b_2, \ldots, \quad b_i, \quad \ldots, b_n) \end{aligned}$$

be vectors such that $\vec{a} \in T$ and $\vec{b} \in F$. For some i and for any $j \in \{1,2,\ldots,n\}$, if there exist a pair of vectors $(\vec{a},\vec{b})$ such that

$$\begin{aligned} &\text{for} \quad j \neq i, \quad a_j = b_j, \quad \text{and} \\ &\text{for} \quad j = i, \quad a_j \neq b_j, \end{aligned}$$

then the function (T,F) **depends on** X_i.

Example 3.1: Let (T,F) be a function shown in Table 2.3. The function depends on X_2, since

$$\begin{aligned} \vec{a}_1 = (1,2,0,1) \in T \quad \text{and} \\ \vec{b}_1 = (1,1,0,1) \in F. \end{aligned}$$

[1]In data mining theory, **support set** is used to mean something else.

The function also depends on X_3, since

$$\vec{a}_2 = (0,1,1,2) \in T \quad \text{and}$$
$$\vec{b}_2 = (0,1,2,2) \in F.$$

■

The following algorithm is an extension of two-valued cases [8], [14], [15] to multi-valued cases [23], [24], [25].

Algorithm 3.1: (Minimization of Variables)

1) For each pair $(\vec{a}, \vec{b})$ such that $\vec{a} \in T$ and $\vec{b} \in F$, 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 $\vec{a} \in T$ and $\vec{b} \in F$, make 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 (T, F) in Table 2.3. This function appears in Examples 2.5 and 3.1.

1) The set of clauses are:

$$\begin{aligned} C(\vec{a}_1, \vec{b}_1) &= y_2, \\ C(\vec{a}_1, \vec{b}_2) &= y_1 \vee y_2 \vee y_3 \vee y_4, \\ C(\vec{a}_2, \vec{b}_1) &= y_1 \vee y_3 \vee y_4, \quad \text{and} \\ C(\vec{a}_2, \vec{b}_2) &= y_3. \end{aligned}$$

2) Forming the product of all the clauses yields

$$R = y_2(y_1 \vee y_2 \vee y_3 \vee y_4)(y_1 \vee y_3 \vee y_4)y_3.$$

3) The simplified expression is $R = y_2 y_3$.
4) The minimum support set is $\{2, 3\}$.

■

Theorem 3.1: If the function f depends on X_i, then any support set of f contains i.

(Proof) If a support set do 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 do not depend on x_i. □

Example 3.3: Consider the function (T, F) shown in Table 2.1. Note that this function appeared in Examples 2.2 and 2.3.

1) The set of clauses are:

$$\begin{aligned} C(\vec{a}_1, \vec{b}_1) &= y_1 \vee y_2 \vee y_3, \\ C(\vec{a}_1, \vec{b}_2) &= y_1 \vee y_3, \\ C(\vec{a}_1, \vec{b}_3) &= y_1 \vee y_2, \\ C(\vec{a}_2, \vec{b}_1) &= y_2 \vee y_3, \\ C(\vec{a}_2, \vec{b}_2) &= y_1 \vee y_2 \vee y_3, \\ C(\vec{a}_2, \vec{b}_3) &= y_1 \vee y_2, \\ C(\vec{a}_3, \vec{b}_1) &= y_2 \vee y_3, \\ C(\vec{a}_3, \vec{b}_2) &= y_1 \vee y_3, \quad \text{and} \\ C(\vec{a}_3, \vec{b}_3) &= y_1 \vee y_2 \vee y_3. \end{aligned}$$

2) By making the product of all the clauses, we have

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

3) The simplified expression is

$$R = y_1 y_2 \vee y_2 y_3 \vee y_1 y_3.$$

4) The minimum support sets are $\{1, 2\}$, $\{2, 3\}$, and $\{1, 3\}$.
5) Thus, any two variables out of the three are necessary to distinguish T from F.

■

IV. Training Data That Should Satisfy Monotone Increasing Property

Assume that the data has the monotone increasing property. However, real data often contain inconsistent instances[2]. Such instances must be removed from the training set,

Theorem 4.1: Let $T, F \subseteq \mathcal{P}^n$ and $T \cup F = \phi$. Then, (T, F) has a monotone increasing extension iff there is no pair $(\vec{a}, \vec{b})$, where $\vec{a} \in T$, $\vec{b} \in F$, and $\vec{a} < \vec{b}$. In such a case (T, F) is consistent.

(Proof) Suppose that (T, F) has a monotone increasing extension (T_1, F_1), then $\vec{a} \geq \vec{b}$ for any $\vec{a} \in T \subseteq T_1$ and $\vec{b} \in F \subseteq F_1$. So, there is no pair $(\vec{a}, \vec{b})$ such that $\vec{a} < \vec{b}$.

If $\vec{a} \in T$ and $\vec{b} \in F$ and $\vec{a} < \vec{b}$, then any extension (T_1, F_1) of (T, F) has the property that $\vec{a} \in T \subseteq T_1$ and $\vec{b} \in F \subseteq F_1$. This means (T, F) has a non-monotone property, and has no monotone extension. □

Example 4.1: Consider the vectors in Table 4.1. Assume that the sets have the monotone increasing property. Then, there should be no pair $(\vec{a}, \vec{b})$ such that $\vec{a} < \vec{b}$.

Note that Table 4.1 contains four conflicting pairs:

$$\begin{aligned} &\vec{a}_3 < \vec{b}_1, \\ &\vec{a}_3 < \vec{b}_2, \\ &\vec{a}_1 < \vec{b}_3, \quad \text{and} \\ &\vec{a}_2 < \vec{b}_3. \end{aligned}$$

We want to remove the fewest vectors to make the sets consistent. How do we do that? If we remove $\vec{a}_3$ and $\vec{b}_3$, then the sets become consistent. ■

Algorithm 4.1: (Removal of Conflicting Instances)

1) If $\vec{a}_i < \vec{b}_j$, then make a clause

$$C(i, j) = a_i \vee b_j.$$

2) For all the clauses, make the product:

$$R = \bigwedge_{(\vec{a}_i,\vec{b}_j)} C(i, j).$$

3) Convert R into the sum-of-products expression.
4) A product with the fewest literals corresponds to a minimum solution.

[2]Data may have some error due to the variations in the measurements obtained by using different instruments in different laboratories.

TABLE 4.1
FUNCTION TABLE INCLUDING INCONSISTENT INSTANCES.

		X_1	X_2	X_3	X_4
T	$\vec{a}_1$	0	0	2	1
	$\vec{a}_2$	0	0	1	2
	$\vec{a}_3$	1	1	0	0
	$\vec{a}_4$	2	2	2	2
F	$\vec{b}_1$	2	1	0	0
	$\vec{b}_2$	1	2	0	0
	$\vec{b}_3$	0	0	2	2
	$\vec{b}_4$	0	0	0	0

For this problem, a more efficient algorithm exists[3]. However, we use Algorithm 4.1, since it uses the same routine as Algorithm 3.1, and also it is fast enough for this problem.

Example 4.2: Let us remove inconsistencies from Table 4.1. This table has four conflict.

1) We have four clauses:
$$a_3 \vee b_1,\ a_3 \vee b_2,\ a_1 \vee b_3,\ a_2 \vee b_3.$$
2) By multiplying the clauses, we have the product:
$$R = (a_3 \vee b_1)(a_3 \vee b_2)(a_1 \vee b_3)(a_2 \vee b_3).$$
3) By converting R into the sum-of-products expression, we have:
$$\begin{aligned} R &= (a_3 \vee b_1 b_2)(a_1 a_2 \vee b_3). \\ &= a_1 a_2 a_3 \vee a_3 b_3 \vee a_1 a_2 b_1 b_2 \vee b_1 b_2 b_3. \end{aligned}$$
4) $a_3 b_3$ is the product with the fewest literals.
5) Thus, the conflicts can be removed by deleting $\vec{a}_3$ and $\vec{b}_3$. This is the minimum solution.

■

Example 4.3: By removing $\vec{a}_3$ and $\vec{b}_3$ from Table 4.1, we have consistent sets. A monotone increasing extension of the resulting table is: $f = X_3^{\{2\}} X_4^{\{1,2\}} \vee X_3^{\{1,2\}} X_4^{\{2\}}$. ■

V. MUSHROOMS

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family [19]. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous class.

The number of the instances is 8124. Among them, 2480 are incomplete, so they were removed. The remaining 5644 instances consists of 3488 edible and 2156 poisonous instances. The number of the variables is 22. Domains of variables range from 2 to 12. The function takes two values: Poisonous (1) and Edible (0). Thus, the function is:

$$\mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_{22} \to \mathcal{B}, \tag{5.1}$$

where $\mathcal{P}_i = \{0, 1, \ldots, p_i - 1\}$ and

$$\begin{array}{lllll} p_1 = 6, & p_2 = 4, & p_3 = 10, & p_4 = 2, & p_5 = 9, \\ p_6 = 4, & p_7 = 3, & p_8 = 2, & p_9 = 12, & p_{10} = 2, \\ p_{11} = 6, & p_{12} = 4, & p_{13} = 4, & p_{14} = 9, & p_{15} = 9, \\ p_{16} = 2, & p_{17} = 4, & p_{18} = 3, & p_{19} = 8, & p_{20} = 9, \\ p_{21} = 6, & p_{22} = 7. & & & \end{array}$$

A. Multi-Valued Method

Our minimization programs found an expression representing poisonous mushrooms consisting of only three variables:

$$Poison = X_5^{\{2,3,4,5,7\}} \vee X_{21}^{\{0,1,2,3,5\}} X_{22}^{\{1,5\}} \vee X_{21}^{\{1,4\}} X_{22}^{\{0,2,5\}},$$

where

- X_5 denotes odor: almond (0), anise (1), creosote (2), fishy (3), foul (4), musty (5), none (6), pungent (7), spicy (8);
- X_{21} denotes population: abundant (0), clustered (1), numerous (2), scattered (3), several (4), solitary (5); and
- X_{22} denotes habitat: grasses (0), leaves (1), meadows (2), paths (3), urban (4), waste (5), woods (6).

This expression covers the all the poisonous mushrooms in the training set, while none of edible ones. However, it also covers mushrooms not in the training set. To minimize the number of variables, we used Algorithm 3.1. To simplify the multiple-valued expression, we used MINI2 [22], which is an improved version of MINI [11].

B. Two-Valued Method

Boros et. al. [5] solved this problem by a two-valued approach. They represented the data set by two-valued variables. In this case, each nominal value of each multi-valued variable was transformed into one two-valued variable that takes value 1 if and only if the original variable takes the chosen nominal value. Thus, the total number of two-valued variables is

$$\sum_{i=1}^{22} p_i = 125,$$

where p_i appeared in (5.1). By minimizing the number of variables, they represented the function by 6 two-valued variables [5]:

$$\begin{array}{lcl} y_4 & : & \text{bruises=no (0)} \\ y_5 & : & \text{odor=none (6)} \\ y_8 & : & \text{gill-size=broad (0)} \\ y_{11} & : & \text{stalk-root=bulbous (0)} \\ y_{20} & : & \text{spore-print-color = white (7)} \\ y_{22} & : & \text{habitat=woods (6)} \end{array}$$

Here,

$$\begin{array}{lll} y_4 = 1 & \text{if and only if} & X_4 = 0, \\ y_5 = 1 & \text{if and only if} & X_5 = 6, \\ \cdots & \cdots & \cdots, \quad \text{and} \\ y_{22} = 1 & \text{if and only if} & X_{22} = 6. \end{array}$$

Note that in the case of multi-valued variables, only three variables are necessary. Also, in the case of two-valued variables, the problem is to find 6 variables out of 125 variables, while in the case of multi-valued variables, the problem is to find 3 variables out of 22 variables. So, we can find an

[3]It can be formulated as a minimum vertex cover problem. Since the graph is bipartite, it can be solved in a polynomial time [3].

exact minimum solution much faster in the case of multi-valued variables. Thus, to distinguish poisonous mushroom, the multi-valued approach is more convenient than the two-valued one.

VI. Hepatitis

This data set has 13 two-valued variables and 6 numerical variables [10]. As for the the numerical variables, 6 to 10 cut points are used to make multi-valued variables.

- X_1 AGE: 10, 20, 30, 40, 50, 60, 70, 80
- X_2 SEX: male, female
- X_3 STEROID: no, yes
- X_4 ANTIVIRALS: no, yes
- X_5 FATIGUE: no, yes
- X_6 MALAISE: no, yes
- X_7 ANOREXIA: no, yes
- X_8 LIVER BIG: no, yes
- X_9 LIVER FIRM: no, yes
- X_{10} SPLEEN PALPABLE: no, yes
- X_{11} SPIDERS: no, yes
- X_{12} ASCITES: no, yes
- X_{13} VARICES: no, yes
- X_{14} BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00
- X_{15} ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
- X_{16} SGOT: 13, 100, 200, 300, 400, 500,
- X_{17} ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
- X_{18} PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
- X_{19} HISTOLOGY: no, yes

The function value is (Die, Live). Incomplete instances are deleted to obtain 80 complete instances: $|T| = 67$, $|F| = 13$, where T corresponds to Live, and F corresponds to Die.

A. Multi-Valued Method

By using Algorithm 3.1, we have the following minimum set:

$$\{X_1, X_{15}, X_{17}, X_{18}\}.$$

B. Two-Valued Method

Boros et. al. solved this problem by a two-valued approach [5]. They obtained the data set with 46 two-valued variables. They minimized the number of variables using two heuristic methods. The first one produced a solution:

$$X_4 = \text{(no, yes)}, \quad X_{11} = \text{(no, yes)}, \quad X_{13} = \text{(no, yes)},$$
$$X_{15} > 120, \quad X_{15} > 200, \quad X_{18} > 50,$$
$$X_{19} = \text{(no, yes)}.$$

The second one produced a solution:

$$X_1 > 40, \quad X_{11} = \text{(no, yes)}, \quad X_{14} > 1.20,$$
$$X_{15} > 120, \quad X_{18} > 40, \quad X_{18} > 50,$$
$$X_{19} = \text{(no, yes)}.$$

The multi-valued approach requires only four variables, while the two-valued approach requires 7 variables. Also, in the case of two-valued variables, the problem is to find 7 variables out of 46 variables, while in the case of multi-valued variables, the problem is to find 4 variables out of 19 variables. So, in the case of multi-valued variables, we can find the exact minimum solution in a short time. Thus, the multi-valued approach is more convenient than the two-valued approach.

VII. Breast Cancer

In this section, as an example of monotone increasing functions, we consider the case of breast cancer. In breast cancer diagnosis, characteristic of individual cells, obtained from a minimally invasive fine needle aspiration, are used to discriminate benign from malignant breast lumps. This allows an accurate diagnosis without the need for the surgical biopsy [17]. The data used for this analysis comes from Dr. William H. Wolberg [16] of the University of Wisconsin Hospitals, Madison [6].

The dataset consists of 699 instances: 458 benign and 241 malignant. It includes 16 incomplete instances, so the remaining 683 are complete. Among the latter, 444 are benign and 239 are malignant. The data has 9 variables, each variable has integer values between 1 and 10. The two-valued output indicates the benign or malignant nature of the tumor. Thus, the function represents $\mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_9 \to \mathcal{B}$, where $\mathcal{P}_i = \{1, 2, \ldots, 10\}$. For each variable, smaller values tend to imply benign, while larger values tend to imply malignant: the decision function is monotone increasing.

A. Multi-Valued Method

In the data set, the number of conflicting pairs is only 18 among all $444 \times 239 = 106116$ pairs [18]. We removed the minimum number of vectors so that the resulting data set became consistent. By using Algorithm 4.1, we made a consistent training set: We removed 2 benign and 4 malignant instance vectors[4]. The resulting data set (i.e., training set) has 677 instances: 442 are benign and 235 are malignant. The minimum number of variables to represent this data set is four: $\{X_1, X_5, X_6, X_8\}$[5]. By simplifying the expressions using monotonicity[22], we had 25 benign and 232 malignant instances. Then, by minimizing the number of variables, we had:

$$\{X_1, X_4, X_5, X_6, X_7, X_8, X_9\}.$$

Thus, when we assume the monotone property, we need to check seven variables out of nine. From this, given an instance, we may decide whether it is benign or malignant. For some cases, we cannot decide, and we describe these as *unclassified*. To minimize the number of variables, we used Algorithm 3.1.

B. Two-Valued Method [5]

Boros et. al. solved this problem by a two-valued approach. To represent a 10-valued variable, 9 two-valued variables are used. That is, for each multi-valued variable, a cut-point is placed between every two consecutive integer values. For example, the two-valued variable "$x_2 \geq 2.5$" takes values 1 if the value of x_2 is equal to or greater than 2.5, and 0 otherwise.

[4]The minimal solution is not unique. Our solution is different from the that of [18], since our algorithm is different.

[5]For unseen instances, the decision using monotone property is not performed.

Thus, in total, the data set is represented by $9 \times 9 = 81$ two-valued variables.

By minimizing the number of variables, they obtained the following results [5]:

$$\begin{array}{lllll} \text{(a)} & X_2 \geq 2.5, & X_6 \geq 3.5, & X_7 \geq 4.5, & X_1 \geq 6.5, \\ & X_3 \geq 4.5, & X_8 \geq 2.5, & X_4 \geq 2.5, & X_6 \geq 8.5, \\ & X_1 \geq 4.5, & X_3 \geq 3.5, & X_5 \geq 5.5. & \\ \text{(b)} & X_1 \geq 4.5, & X_1 \geq 6.5, & X_2 \geq 2.5, & X_3 \geq 3.5, \\ & X_3 \geq 4.5, & X_4 \geq 1.5, & X_4 \geq 2.5, & X_5 \geq 3.5, \\ & X_6 \geq 3.5, & X_6 \geq 7.5, & X_7 \geq 4.5, & X_8 \geq 2.5. \end{array}$$

Note that (a) requires 11 two-valued variables, while (b) requires 12 two-valued variables. The multi-valued approach requires only seven variables. Also, in the case of two-valued variables, the problem is to find 11 variables out of 81 variables, while in the case of multi-valued variables, the problem is to find 7 variables out of 9 variables. Thus, in the case of multi-valued variables, we can obtain an exact minimum solution in a short time. Thus, the multi-valued approach is more convenient than the two-valued one.

VIII. Conclusion and Comments

This paper presents a method to minimize the number of variables for multi-valued input partially defined decision functions. Major contributions are as follows: We

- Minimized the number of variables for a poisonous mushroom data set; hepatitis data set, and breast cancer data set.
- Compared the multi-valued approach with the two-valued approach, and showed the usefulness of the multi-valued approach.

In the past, these problems were solved by two-valued tools [4], [9], [27]. However, for the functions with multi-valued inputs, the direct reduction of multi-valued variables is more convenient and efficient.

For medical applications, the method must be understandable and explainable. Our method can provide simple reasons for the decision. Our explanation should be obvious for doctors, patients, hospitals, insurance companies, the pharmaceutical industry, and various government agencies involved in health care [9]. This is different from other methods such as neural nets, where the explanation for the decision is hard.

Acknowledgments

This research is partly supported by the Japan Society for the Promotion of Science (JSPS) Grant in Aid for Scientific Research. Discussion with Prof. Jon T. Butler was quite useful.

References

[1] D. Angluin and C. H. Smith, "Inductive inference: Theory and methods," *Computing Surveys,* Vol. 15, No. 3, Sept. 1983, pp. 327-369.

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

[3] E. Boros, T. Ibaraki, and K. Makino, "Error-free and best-fit extensions of partially defined Boolean functions," *Information and Computation*, Vol. 140, pp. 254-283, 1998.

[4] 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.

[5] 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.

[6] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+

[7] Y. Crama, P. L. Hammer and T. Ibaraki, "Cause-effect relationships and partially defined Boolean functions," *Annals of Operations Research* Vol. 16 ,1988, pp. 299-326.

[8] C. Halatsis and N. Gaitanis, "Irredundant normal forms and minimal dependence sets of a Boolean functions," *IEEE Trans. on Computers*, vol. C-27, no. 11, Nov. 1978, pp. 1064-1068.

[9] 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.

[10] https://archive.ics.uci.edu/ml/datasets/hepatitis

[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] T. Ibaraki,"Partially defined Boolean functions," Chapter 8 in: Y. Crama and P. L. Hammer, *Boolean Functions - Theory, Algorithms and Applications*, Cambridge University Press, New York, 2011.

[14] Y. Kambayashi, "Logic design of programmable logic arrays," *IEEE Trans. on Computers*, vol. C-28, no. 9, Sept. 1979, pp. 609-617.

[15] J. Kuntzmann, *Algèbre de Boole*, Dunod, Paris, 1965. English translation: *Fundamental Boolean Algebra*, Blackie and Son Limited, London and Glasgow, 1967.

[16] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, Vol. 23, No. 5, Sept.1990, pp. 1-18.

[17] O. L. Mangasarian, W. N. Street, and and W. H. Wolberg, "Breast cancer diagnosis via linear programming," *Mathematical Programming Technical Report 94-10*, Dec. 19, 1994.

[18] K. Makino, T. Suda, H. Ono, and T. Ibaraki, "Data analysis by positive decision trees," *IEICE Trans. Inf. & Syst.*, Vol E82-D, No.1 Jan 1999, pp. 76-88.

[19] https://archive.ics.uci.edu/ml/datasets/mushroom

[20] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, Vol. 1, pp. 81-106, 1986.

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

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

[23] T. Sasao, "On the number of dependent variables for incompletely specified multiple-valued functions," *International Symposium on Multiple-Valued Logic (ISMVL-2000)*, Portland, OR, USA, pp. 91-97, May, 2000.

[24] T. Sasao, "On the numbers of variables to represent sparse logic functions,"*International Conference on Computer Aided Design (ICCAD-2008)*, pp. 45-51, November, 2008.

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

[26] T. Sasao, "Index generation functions: Tutorial," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 23, No. 3-4, pp. 235-263, 2014.

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