Linear Decompositions for Multi-Valued Input Classification Functions

Tsutomu Sasao
Meiji University,
Kawasaki, 214-8571 Japan
sasao@ieee.org

Jon T. Butler
Naval Postgraduate School
Monterey, CA, 93943 USA
jon_butler@msn.com

***Abstract*—In a multi-valued input classification function, each input combination represents properties of an object, while the output represents the class of the object. Each variable may have different radix. In most cases, the functions are partially defined. To represent multi-valued variables, both one-hot and minimum-length encoding are considered. Experimental results using University of California Irvine (UCI) benchmark functions show that the one-hot approach results in fewer variables than the minimum-length approach with linear decompositions.**

I. Introduction

We consider data sets with multi-valued inputs. Various methods exist to represent a multi-valued variable. To keep the original structure of the data, we use one-hot encoding. That is, to represent a q-valued variable, we use q binary variables. Although this increases the total number of variables temporarily, the number of variables can be reduced by a linear decomposition. The original functions can be decomposed into two parts: a linear part, and a general part, as shown in Fig. 1.1. With this technique, many University of California Irvine (UCI) data sets for machine learning [21] were successfully decomposed. In contrast, conventional functional decompositions [1], [3], [9], [12] are more constrained and less likely to simplify. The rest of the paper is organized as follows: Section II shows definitions and basic properties; Section III introduces linear decomposition; Section IV shows an illustrative example; Section V shows experimental results using UCI data sets; Section VI surveys related works; and Section VII concludes the paper.

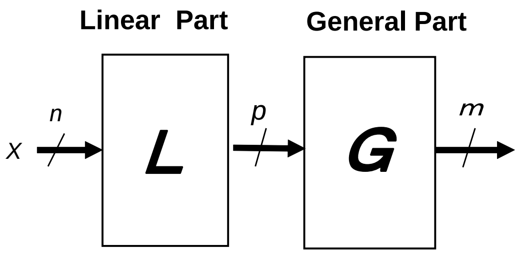

Fig. 1.1. Linear Decomposition

II. Definitions and Basic Properties

Definition 2.1: A **multi-valued partially defined classification function** is a mapping $f : D \to M$, where $D \subset Q^N$, $Q = \{0, 1, 2, \dots, q-1\}$, $M = \{1, 2, \dots, m\}$, and N is the number of multi-valued variables. Each element of D is called a **registered vector**. There are k such vectors. k is called the **weight** of the function. For each registered vector, assign an integer between 1 and m, where $2 \leq m \leq k$. A **registered vector table** shows the **function value** for each registered vector. A partially defined classification function f produces the corresponding function value when the input vector matches a registered vector. Let F_i be the set of registered vectors which map to $i \in M$. Then, $D = \bigcup_{i=1}^{m} F_i$. We assume that $F_i \neq \varnothing$ $(i = 1, 2, \dots, m)$.

TABLE 2.1
Registered Vector Table

x_1	x_2	x_3	x_4	x_5	x_6	f
1	1	0	0	1	1	1
0	1	1	0	1	1	1
0	1	0	1	0	0	2
0	0	0	0	1	0	2
1	1	0	1	1	1	3
1	0	0	0	1	1	3

Example 2.1: The registered vector table in Table 2.1 shows a classification function with $N = 6$, $m = 3$ and $k = 6$. Here, the input variables are binary. ■

Definition 2.2: [4] For $F_i, F_j \subset Q^N$, $(i, j = 1, 2, \dots, m)$, and $i \neq j$, let $F_i \cap F_j = \varnothing$. An m-tuple $(F_1, F_2, \dots, F_m)$ denotes a partially defined classification function when $\bigcup_{i=1}^{m} F_i \subset Q^N$. For a partially defined function $(F_1, F_2, \dots, F_m)$, the function $(E_1, E_2, \dots, E_m)$ that satisfies $F_i \subseteq E_i \subset Q^N$ is an **extension** of $(F_1, F_2, \dots, F_m)$, where $E_i \cap E_j = \varnothing$ $(i \neq j)$.

Definition 2.3: For a subset $U \subseteq Q^N$ and $S \subseteq \{1, 2, \dots, N\}$, we denote by $U|_S$ the **projection** of U to S. In other words, $U|_S = \{\vec{a}|_S\}$, where $\vec{a}|_S$ is obtained from $\vec{a} \in U$ by considering only those components a_j with $j \in S$.

Example 2.2: Let $Q = \{0, 1, 2\}$ and $N = 4$. Let $U = \{(1, 2, 0, 1), (0, 1, 1, 2), (2, 0, 1, 2)\}$ and $S = \{2, 3\}$. Then, we have the projection $U|_S = \{(*, 2, 0, *), (*, 1, 1, *), (*, 0, 1, *)\}$. ■

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 $F_i \subset Q^N$ $(i = 1, 2, \dots, m)$. Given a partially defined function $(F_1, F_2, \dots, F_m)$, and a subset $S \subseteq \{1, 2, \dots N\}$, if $F_i|_S \cap F_j|_S = \varnothing$, $(i \neq j)$ holds, then S

TABLE 2.2
CLASSIFICATION FUNCTION WITH REDUCED VARIABLES

x_1	x_2	x_4	f	TAG	$y_1 = x_1$	$y_2 = x_2 \oplus x_4$
1	1	0	1	1	1	1
0	1	0	1	2	0	1
0	1	1	2	3	0	0
0	0	0	2	4	0	0
1	1	1	3	5	1	0
1	0	0	3	6	1	0

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, $j \in \{1, 2, \ldots, n\} - S$, and the variable is **redundant**.

Example 2.3: Consider the function (F_1, F_2, F_3) shown in Table 2.1. In this case, $S = \{1, 2, 4\}$ is a support set, since for

$$\begin{aligned} F_1|_S &= \{(1,1,*,0,*,*), (0,1,*,0,*,*)\}, \\ F_2|_S &= \{(0,1,*,1,*,*), (0,0,*,0,*,*)\}, \\ F_3|_S &= \{(1,1,*,1,*,*), (1,0,*,0,*,*)\}, \end{aligned}$$

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

Algorithms to represent a given function by using the minimum number of variables have been developed [11].

III. LINEAR DECOMPOSITION

In this section, we assume that the input variables are binary. Let n be the number of binary variables.

The number of variables of a partially defined classification function $f : D \to M$, where $D \subset B^n$ and $B = \{0, 1\}$ often can be reduced by a **linear decomposition** [6], [8], [18]. In the linear decomposition shown in Fig. 1.1, L realizes a linear function, while G realizes a general function (in most cases, a non-linear function).

Definition 3.1: A **compound variable** has the form $y = c_1x_1 \oplus c_2x_2 \oplus \cdots \oplus c_nx_n$, where $c_i \in \{0, 1\}$. The **compound degree** of a variable y is $\sum_{i=1}^{n} c_i$, where $\sum$ denotes ordinary integer addition, and c_i is treated as an integer. A **primitive variable** is a variable whose compound degree is one.

When a partially defined function satisfies a certain condition, there exists a linear transformation that reduces the number of variables p.

Definition 3.2: In a partially defined function $(F_1, F_2, \ldots, F_m)$, let $\vec{a} \in F_i$, and $\vec{b} \in F_j$, $(i \neq j)$. Then, the vector $\vec{d} = \vec{a} \oplus \vec{b}$ is a **difference vector**. The set of the difference vectors is denoted by D_f.

Lemma 3.1: [20] An n-variable partially defined classification function f can be represented with $n - 1$ compound variables if and only if there exists a non-zero vector $\vec{u}$ such that $\vec{u} \in B^n - D_f$, where D_f is the set of difference vectors of f.

Theorem 3.1: [20] An n-variable partially defined classification function f can be represented with at most $n - 1$ compound variables if and only if the set of difference vectors D_f contains less than $2^n - 1$ distinct difference vectors.

Algorithm 3.1: (Reduction of Compound Variables [19])

TABLE 3.1
THE SET OF DIFFERENCE VECTORS FOR THE FUNCTION IN TABLE 2.2

x_1	x_2	x_4	Vector Pairs
0	0	1	$(1,5), (2,3)$
0	1	0	$(1,6), (2,4)$
1	0	0	$(3,5), (4,6)$
1	0	1	$(1,3), (2,5)$
1	1	0	$(1,4), (2,6)$
1	1	1	$(3,6), (4,5)$

TABLE 3.2
CLASSIFICATION FUNCTION AFTER LINEAR TRANSFORMATION

y_1	y_2	f
1	1	1
0	1	1
0	0	2
0	0	2
1	0	3
1	0	3

1) Derive the set of difference vectors D_f of an n-variable function.
2) If $|D_f| = 2^n - 1$, then stop, since reduction is impossible.
3) Obtain a non-zero vector $\vec{d} \in B^n - D_f$ with minimum weight.
4) Remove one variable from $\vec{d}$, and apply the linear transformation to the function.
5) Let $n \leftarrow n - 1$, and go to step 1.

Example 3.1: Let us reduce the number of variables in the classification function shown in Table 2.2.

1) Table 3.1 shows the set of difference vectors. The rightmost column shows the pairs of registered vectors that produced the difference vectors. 12 difference vectors were generated, but only 6 vectors are distinct.
2) $|D_f| = 6 < 2^3 - 1$.
3) In this case, the set of difference vectors does not contain $\vec{d} = (0, 1, 1)$. We select $\vec{d} = (0, 1, 1)$.
4) Perform the linear transformation that converts (x_2, x_4) into $y_2 = x_2 \oplus x_4$. Thus, Table 2.2 is converted into Table 3.2, where $y_1 = x_1$.
5) In this case, the function can be represented with two variables: $y_1 = x_1$ and $y_2 = x_2 \oplus x_4$. The input columns in Table 3.2 show that y_1 and y_2 distinguish the three function values.

■

An efficient linear decomposition algorithm has been developed [19]. With this algorithm, functions with more than one thousand inputs have been successfully decomposed.

IV. ILLUSTRATIVE EXAMPLE

Our method to find multi-valued decompositions consists of two steps:

1) Represent the function using a binary encoding.
2) Perform a linear decomposition.

To show the method, consider the 2-variable 3-valued function, where the variables are 5-valued, as shown in Table 4.1.

TABLE 4.1
5-VALUED INPUT 3-VALUED OUTPUT FUNCTION.

X_1	X_2	f
2	4	0
0	3	0
0	1	0
3	1	1
2	3	1
2	2	1
4	0	2
3	3	2
1	4	2

A. *One-hot Encoding*

In one-hot representations, 0, 1, 2, 3, 4, and 5 become $(1,0,0,0,0)$, $(0,1,0,0,0)$, $(0,0,1,0,0)$, $(0,0,0,1,0)$, and $(0,0,0,0,1)$, respectively. Similarly, the three output values for f, 0, 1, and 2, become $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$, respectively. With this substitution, Table 4.1 becomes Table 4.2.

TABLE 4.2
ONE-HOT ENCODED FUNCTION.

x_0	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3	y_4	z_0	z_1	z_2
0	0	1	0	0	0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	0	1	0	1	0	0
1	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0	0	1

The function in Table 4.2 is represented algebraically as

$$Z = f(X, Y),$$

where $X = (x_0, x_1, x_2, x_3, x_4)$, $Y = (y_0, y_1, y_2, y_3, y_4)$, and $Z = (z_0, z_1, z_2)$. This function has a sum-of-products expression

$$\begin{aligned} z_0 &= x_2y_4 \vee x_0y_3 \vee x_0y_1, \\ z_1 &= x_3y_1 \vee x_2y_3 \vee x_2y_2, \\ z_2 &= x_4y_0 \vee x_3y_3 \vee x_1y_4. \end{aligned} \tag{4.1}$$

Using Algorithm 3.1 yields the linear transformation:

$$\begin{aligned} w_0 &= x_3 \oplus y_2 \oplus y_3, \\ w_1 &= x_2 \oplus y_1. \end{aligned}$$

In this case, $Z = (z_0, z_1, z_2)$ can be represented by only two variables, as shown in Table 4.3. Thus, the function can be represented by the following equations.

$$\begin{aligned} z_0 &= \bar{w}_0w_1 \vee w_0\bar{w}_1, \\ z_1 &= w_0w_1, \\ z_2 &= \bar{w}_0\bar{w}_1. \end{aligned} \tag{4.2}$$

Therefore, only two variables, w_0 and w_1, are needed between L and G in Fig. 1.1.

TABLE 4.3
ONE-HOT ENCODED FUNCTION: AFTER LINEAR TRANSFORMATION

w_0	w_1	z_0	z_1	z_2
0	1	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	0	1	0
1	1	0	1	0
1	1	0	1	0
0	0	0	0	1
0	0	0	0	1
0	0	0	0	1

B. *Minimum-length Encoding*

In minimum-length encoding, a q-valued variable is represented by $\lceil \log_2 q \rceil$ bits. We assume that the input part is represented by minimum-length encoding, while the output part is represented by one-hot encoding. Thus, Table 4.1 becomes Table 4.4. In this case, the number of variables can be reduced to four variables: x_1, x_0, y_1, and y_0. Table 4.5 shows the function.

TABLE 4.4
MINIMUM-LENGTH ENCODED FUNCTION.

x_2	x_1	x_0	y_2	y_1	y_0	z_0	z_1	z_2
0	1	0	1	0	0	1	0	0
0	0	0	0	1	1	1	0	0
0	0	0	0	0	1	1	0	0
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	0	1	0
0	1	0	0	1	0	0	1	0
1	0	0	0	0	0	0	0	1
0	1	1	0	1	1	0	0	1
0	0	1	1	0	0	0	0	1

TABLE 4.5
MINIMUM-LENGTH ENCODED FUNCTION:AFTER REDUCTION OF VARIABLES

x_1	x_0	y_1	y_0	z_0	z_1	z_2
1	0	0	0	1	0	0
0	0	1	1	1	0	0
0	0	0	1	1	0	0
1	1	0	1	0	1	0
1	0	1	1	0	1	0
1	0	1	0	0	1	0
0	0	0	0	0	0	1
1	1	1	1	0	0	1
0	1	0	0	0	0	1

Algorithm 3.1 further reduced the number of variables to three as shown in Table 4.6, where $w = x_0 \oplus y_1$.

TABLE 4.6
MINIMUM-LENGTH ENCODED FUNCTION:AFTER LINEAR TRANSFORMATION

x_1	w	y_0	z_0	z_1	z_2
1	0	0	1	0	0
0	1	1	1	0	0
0	0	1	1	0	0
1	1	1	0	1	0
1	1	1	0	1	0
1	1	0	0	1	0
0	0	0	0	0	1
1	0	1	0	0	1
0	1	0	0	0	1

V. Experimental Results

Table 5.1 summarizes the benchmark functions used in the experiments. N is the original number of variables; m is the number of classes; k is the number of instances, and r is the radix. Original data were taken from the UCI (University of California, Irvine) machine learning repository [21].

Both one-hot encoding and minimum-length encoding were considered. First, original data were represented by one-hot encoding or minimum length encoding. Second, the linear decomposition algorithm in [19] was used to reduce the number of variables. Table 5.2 shows the experimental results. n is the number of bits; p_0 is the number of bits after removing redundant bits; and p is the number of bits after linear decomposition. The superior approach is indicated by a bold entry in Table 5.2. For these benchmark functions, one-hot encodings tend to produce smaller p. Over all 17 benchmark functions, the one-hot approach was superior to the minimum-length approach in 8 cases, while the minimum length approach was superior in one case. In 8 cases, both approaches produced the same number p of variables.

The most time-consuming problem was Connect-4. It took 5709 seconds to reduce 126 variables into 22 variables using one-hot encoding. We used a computer with an Intel Core i7 7700 CPU, 4 cores, with 64GB main memory, on Windows 10.

Brief descriptions of benchmark functions follow. Details are available in [21].

TABLE 5.1
Function Data [21]

Data Name	N	m	k	r
Tic Tac Toe	9	3	958	3
Connect-4	42	3	67557	3
Chess3196	36	2	3196	$2, 3, 4$
Chess28056	6	18	28056	$4, 8$
Poker	10	10	25010	$4, 13$
Balance	4	3	625	5
MONK's	6	2	432	$2, 3, 4$
Lymphography	18	4	147	$2, 3, 4, 8$
Breast Cancer	9	2	677	10
Hepatitis	19	2	80	$2 \sim 11$
Nursery Schools	8	5	12960	$2, 3, 4, 5$
Car Evaluation	6	4	1728	$3, 4$
Zoo	16	7	101	$2, 6$
Vote	16	2	435	3
Promoter	57	2	106	4
Splice	60	3	3174	4
Mushrooms	22	2	5644	$2 \sim 12$

N: Original number of variables m: Number of classes
k: Number of instances r: Radix

A. *Tic-Tac-Toe*

In tic-tac-toe, two players, X and Y, take turns marking the spaces in a 3×3 grid shown in Fig. 5.1. The player who first succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner [22]. This dataset encodes the complete set of possible board configurations at the end of a tic-tac-toe game, where X is assumed to have played first. The target concept is "win for X", i.e. true when X has one of 8 possible ways to create a "three-in-a-row". The number of instances is $k = 958$ (legal tic-tac-toe endgame boards). The number of variables is $N = 9$, and each variable takes $q = 3$ values: (0) player X occupies, (1) player Y occupies, and (2) empty. The function takes $m = 3$ classes[1] : (0) player X wins, (1) player Y wins, and (2) draw. Among 958 instances, player X wins for 626 instances, player Y wins for 316 instances, and a draw occurs for 16 instances.

TABLE 5.2
Comparison of Encodings.

	One-hot			Minimum-length		
Data Name	n	p_0	p	n	p_0	p
Tic Tac Toe	27	9	9	18	12	9
Connect-4	126	61	22	84	63	22
Chess3196	75	30	15	38	30	15
Chess28056	40	34	16	16	16	16
Poker	85	61	**21**	30	26	23
Balance	20	16	**11**	12	12	12
MONK's	17	5	**3**	10	8	4
Lymphography	59	11	**8**	29	11	9
Breast Cancer	90	13	10	36	11	**9**
Hepatitis	67	7	**5**	32	7	6
Nursery Schools	27	17	**14**	16	16	15
Car Evaluation	21	14	12	12	12	12
Zoo	36	6	5	19	7	5
Vote	48	11	8	32	11	8
Promoter	228	6	**6**	114	7	7
Splice	240	18	**15**	120	18	17
Mushrooms	118	5	4	53	7	4

n : Number of bits
p_0 : Number of bits after removing blueundant bits
p : Number of bits after linear decomposition

The function corresponds to a mapping: $D \rightarrow \{0, 1, 2\}$, where $D \subset \{0, 1, 2\}^9$, and $|D| = 958$.

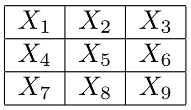

Fig. 5.1. Board for tic-tac-toe

With one-hot encoding, this function was converted to a function with $n = N \times q = 27$ binary variables. n corresponds to the number of bits to represent the input part of the function. The number of primitive variables is reduced to $p_0 = 9$ variables.

B. *Connect-4*

Connect-4 is a two-player board game. Two players, X and Y, drop disks into a vertically suspended 6×7 grid shown in Fig. 5.2. The disks fall straight down, occupying the lowest available space within the columns. The objective of the game is to be the first to form a a horizontal, vertical, or diagonal line of four of one's own disks.

This dataset contains all legal 8-ply (8 turn) positions in the game of Connect-4 in which neither player has won, and in which the next move is not forced. The number of variables is $N = 6 \times 7 = 42$, since there are 6 rows and 7 columns in the grid. Each variable takes $q = 3$ values, empty, X, and Y. The number of classes is $m = 3$: (0) player X wins, (1) player Y wins, and (2) draw. The number of instances is

[1]The function does not show how to win. Instead, it represents a win for X, a win for Y or, a win for neither, assuming each player plays optimally.

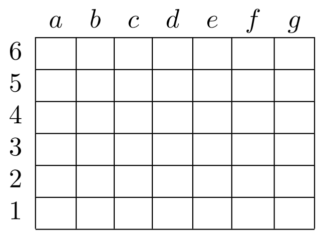

Fig. 5.2. Board for Connect-4

$k = 67557$. Among these, player X wins for 44473 instances, player Y wins for 16635 instances, and a draw occurs for 6449 instances. The number of bits to represent the input part is $n = 42 \times 3 = 126$. The number of primitive variables is reduced to $p_0 = 61$, and a linear decomposition reduced the number of compound variables to $p = 22$,

C. Chess3196

Chess3196 is chess with $k = 3196$ starting positions. The number of classes is $m = 2$: (0) white can win, and (1) white cannot win. The number of variables is $N = 36$: 34 of them take two values; one of them takes three values; and one of them takes four values. Thus, the total number of binary variables is $n = (34 \times 2) + (1 \times 3) + (1 \times 4) = 75$.

D. Chess28056

Chess28056 is chess with $k = 28056$ starting positions. The number of classes is $m = 18$. It shows the optimal depth-of-win for White in 0 to 16 moves, otherwise draw. The number of variables is $N = 6$: two of which take four values; and four of which take eight values. Thus, the total number of binary variables is $n = (2 \times 4) + (4 \times 8) = 40$.

E. Poker

Each hand consists of five playing cards drawn from a standard deck of 52. Each card is described by two attributes (suit and rank). Thus, the number of variables is $N = 10$: five of which take four values (Hearts, Spades, Diamonds, Clubs); and five of which take 13 values (Ace, 2, 3, ... , Queen, King). Thus, the total number of binary variables is $n = (5 \times 4) + (5 \times 13) = 85$. The class denotes the poker hand, and the number of classes is $m = 10$: (1) Nothing, (2) One pair, (3) Two pairs, (4) Three of a kind, (5) Straight, (6) Flush, (7) Full house, (8) Four of a kind, (9) Straight flush, and (10) Royal flush. The number of instances is $k = 25010$.

F. Balance Scale

This data models the operation of a balance scale. The number of classes is $m = 3$: (1) tip on the right, (2) balanced, and (3) tip on the left. $N = 4$ variables are used: X_1 (the left weight), X_2 (the left distance), X_3 (the right weight), and X_4 (the right distance). Each variable takes one of $q = 5$ values: $1, 2, 3, 4, 5$. The total number of binary variables is $n = 4 \times 5 = 20$. The instances completely cover the attribute space. Thus, the number of instances is $k = 5^4 = 625$. The class is represented by

$$f = \begin{cases} 1 & \text{When } (X_1 \times X_2) > (X_3 \times X_4) \\ 2 & \text{When } (X_1 \times X_2) = (X_3 \times X_4) \\ 3 & \text{When } (X_1 \times X_2) < (X_3 \times X_4) \end{cases}$$

G. MONK's Problem

The MONK's problems were the basis of the first international comparison of learning algorithms. The number of classes is $m = 2$. The number of variables is $N = 6$: two of them take two values; three of them take three values; and one of them takes four values. Thus, the total number of binary variables is $n = (2 \times 2) + (3 \times 3) + (1 \times 4) = 17$. The instances completely cover the attribute space. Thus, the number of instances is $k = 2^2 \times 3^3 \times 4 = 432$.

H. Lymphography

This data came from the University Medical Center, Institute of Oncology, Ljubljana, Slovenia in Nov. 1988. The number of instances is $k = 148$. The number of classes is $m = 4$: normal, metastases, malignant lymph, and fibrosis. The number of variables is $N = 18$. Nine of them take two values; three of them take three values; four of them take four values; and one of them takes eight values. Thus, the total number of binary variables is $n = (9 \times 2) + (3 \times 3) + (4 \times 4) + (1 \times 8) = 59$.

I. Breast Cancer

This data came from the Clinical Sciences Center, University of Wisconsin. Features are computed from a digitized image of a fine needle aspiration (FNA) of a breast mass. The original dataset consists of 699 instances, including 16 incomplete instances, which are removed. We also removed the minimum number of instances so that the resulting dataset became consistent [17]. The number of remaining instances is $k = 677$. The number of classes is $m = 2$: (0) malignant, and (1) benign. The number of variables is $N = 9$, and each takes $q = 10$ values. Thus, the total number of binary variables is $n = 9 \times 10 = 90$.

J. Hepatitis

This data came from Ljubljana, Slovenia in Nov. 1988. The number of instances is $k = 80$. The number of classes is $m = 2$: (0) die and (1) live. The original dataset has 13 two-valued variables and 6 numerical variables. As for the numerical variables, 6 to 10 cut points are used to make multi-valued variables. Thus, the resulting number of variables is $N = 19$: 13 of which take two values; two of which take seven values; two of which take five values; one of which takes 6 values; and one of which takes 11 values. Thus, the total number of binary variables is $n = (13 \times 2) + (2 \times 7) + (2 \times 5) + (1 \times 6) + (1 \times 11) = 67$.

K. Nursery Schools

A nursery school dataset was used to rank applications for nursery schools. It was used during 1980's when there was excessive enrollment in these schools in Ljubljana, Slovenia, since rejected applications frequently needed an objective explanation. The number of classes is $m = 5$: not recommended, recommended, very recommended, priority, special priority. The number of variables is $N = 8$: one of which takes two values; four of which take three values; two of which take four values; and one of which takes five values. Thus, the total number of binary variables is $n = (1 \times 2) + (4 \times 3) + (2 \times 4) + (1 \times 5) = 27$. The instances completely cover the attribute space. Thus, the number of instances is $k = 2 \times 3^4 \times 4^2 \times 5 = 12960$.

L. Car Evaluation

This data model for evaluating cars based on their price and technical characteristics. The number of classes is $m = 4$: (0) unacceptable, (1) acceptable, (2) good, (3) very good. The number of variables is $N = 6$: three of which take three values; and three of which take four values. Thus, the total number of binary variables is $n = (3 \times 3) + (3 \times 4) = 21$. The instances completely cover the attribute space. Thus, the number of instances is $k = 3^3 \times 4^3 = 1728$.

M. Zoo

Given the properties of animals, this function shows the animal class. The number of classes is $m = 7$: (1) Mammals, (2) Birds, (3) Reptiles, (4) Fish, (5) Amphibians, (6) Insects, (7) Others. Properties of animals include: whether it drinks milk when it is a baby, whether it has feathers or not, number of legs etc. The number of variables is $N = 16$: 15 of them take two values, and one of them takes 6 values. Thus, the total number of binary variables is $n = (15 \times 2) + (1 \times 6) = 36$. The number of instances is $k = 101$.

N. Vote

This data is from the 1984 United States Congressional Voting Records. The number of instances, that is equal to the number of representatives, is $k = 435$. The number of classes is $m = 2$: (0) democrat, and (1) republican. The number of variables is $N = 16$; each takes three values: (0) yes, (1) no, (2) other. Thus, the total number of binary variables is $n = 16 \times 3 = 48$.

O. Promoter

This data is related to gene sequences (DNA), where the number of instances is $k = 106$. The number of classes is $m = 2$: positive or negative. The number of variables is $N = 57$, each takes four values: (1) A, (2) G, (3) T, and (4) C. Thus, the total number of binary variables is $n = 57 \times 4 = 228$.

P. Splice

This data is related to gene sequences (DNA). Originally, the data consists of 3191 instances. Removing 16 instances that had ambiguous inputs (D, N, S, R), resulted in a data set with $k = 3174$ instances. The number of classes is $m = 3$: (0) donor, (1) acceptor, (2) neither. The number of variables is $N = 60$; each takes four values: (1) A, (2) G, (3) T, and (4) C. Thus, the total number of binary variables is $n = 60 \times 4 = 240$.

Q. Mushroom

This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. 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 original data contains 8124 mushrooms; among them, 2480 are incomplete, which are removed. Thus, the number of instances is $k = 5644$. The number of classes is $m = 2$: (1) edible, and (2) not edible. The number of variables is $N = 22$. Among them, four variables take two values; two take three values; five take four values; three take six values; one takes seven values; one takes eight values; four take nine values; one takes 10 values; and one takes 12 values. Thus, the total number of binary variables is $n = (4 \times 2) + (2 \times 3) + (5 \times 4) + (3 \times 6) + (1 \times 7) + (1 \times 8) + (4 \times 9) + (1 \times 10) + (1 \times 12) = 125$. Since, we removed incomplete data, the number of binary variables was reduced to 118.

VI. Related Works

Decompositions of multi-valued input functions In [23], many multi-valued benchmark functions were decomposed to show the usefulness for machine learning. However, they did not show the numbers of bits after decompositions in the table. But, they did show that the numbers of reduced attributes (multi-valued variables) for MONK's, vote, splice, and mushrooms are 3, 10, 12, and 5, respectively.

In [5], bi-decompositions of multi-valued functions were considered. In [16], an efficient optimum functional decompositions algorithm was developed for index generation functions. In [14], [15], linear decompositions of multi-valued input index generation functions were considered, where $mod(p)$ additions were used in the linear transformations.

Support reduction of multi-valued input functions In [7], BDDs were used to reduce the support of binary encoded multi-valued functions. This is the most relevant work to our research. They did not use the UCI data set, but used the POLO benchmark set, which is not available now. So, we cannot check if the data are the same. They used minimum-length encoding to represent multi-valued input and output values. On the other hand, we used one-hot encoding to represent the output values. So, direct comparison is difficult. As for zoo and mushrooms, the results are the same. As for tic-tac-toe, they obtained an $p = 8$-bit solution.

In [2], one-hot encodings were used to find essential variables. In [17], variable minimization of multi-valued input binary functions (i.e., $m = 2$) was considered.

VII. Conclusion and Comments

This paper shows a method to decompose multi-valued input classification functions. First, original functions were converted into partially defined functions. Then, linear decompositions were used to reduce the number of variables. With this strategy, functions with many variables were decomposed successfully. Both one-hot encoding and minimum-length encoding were considered. Over all 17 benchmark functions, the one-hot approach was superior to the minimum-length approach in 8 cases, while the minimum length approach was superior in one case. In 8 cases, both approaches produced the same number p of variables. The superior approach is indicated by a bold entry in Table 5.2.

In a one-hot encoding, a function has more variables in the initial representation. This means that more choices exist for a linear transformation to reduce the variables. On the other hand, in a minimum-length encoding, fewer choices exist, and the number of applicable linear transformations is smaller. Thus, the one-hot encodings tend to result in fewer variables. This result is consistent with the experimental results in index generation functions [18]. That is, when the number of registered vectors are the same, randomly generated functions with more variables in the original representations tend to result in a fewer variables by linear transformations.

This method is promising for data mining.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. We thank three referees whose suggestions improved this paper.

REFERENCES

[1] R. L. Ashenhurst, "The decomposition of switching functions," *International Symposium on the Theory of Switching*, pp. 74-116, April 1957.

[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] H. A. Curtis, *A New Approach to the Design of Switching Circuits*, D. Van Nostrand Co., Princeton, NJ, 1962.

[4] 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.

[5] C. Lang and B. Steinbach, "Bi-decomposition of function sets in multiple-valued logic for circuit design and data mining," In: *Artificial Intelligence Reivew*, Vol. 20, pp. 233-267, Springer, Dordrecht, Dec. 2003.

[6] R. J. Lechner, "Harmonic analysis of switching functions," in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[7] A. Mishchenko, C. Files, M. Perkowski, B. Steinbach, and Ch. Dorotska, "Implicit algorithms for multi-valued input support manipulation,", *Proc. 4th Intl. Workshop on Boolean Problems*, Sept. 2000, Freiberg, Germany.

[8] E. I. Nechiporuk, "On the synthesis of networks using linear transformations of variables," *Dokl. AN SSSR*, vol. 123, no. 4, Dec. 1958, pp. 610-612 (in Russian).

[9] T. Sasao, "Totally undecomposable functions: applications to efficient multiple-valued decompositions," *ISMVL-1999*, Freiburg, Germany, May 20-23, 1999, pp. 59-65.

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

[11] T. Sasao, "On the number of variables to represent sparse logic functions," *ICCAD-2008*, San Jose, California, USA, Nov. 10-13, 2008, pp. 45-51.

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

[13] T. Sasao, "Linear decomposition of index generation functions," *17th Asia and South Pacific Design Automation Conference* (ASPDAC-2012), 2012, Sydney, Australia, pp. 781-788.

[14] T. Sasao, "Multiple-valued input index generation functions: Optimization by linear transformation," *ISMVL-2012*, Victoria, Canada, May 14-16, 2012. pp. 185-190.

[15] T. Sasao, "Multiple-valued index generation functions: Reduction of variables by linear transformation," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 21, No. 5-6, pp. 541-559, 2013.

[16] T. Sasao, K. Matsuura, Y. Iguchi, "An algorithm to find optimum support-reducing decompositions for index generation functions." *DATE-2017*, March 27-31, 2017, Lausanne, Switzerland, pp. 812-817.

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

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

[19] T. Sasao, "Reduction methods of variables for large-scale classification functions,"*International Workshop on Logic and Synthesis* (IWLS-2020), July 27-29, 2020, pp. 82-87.

[20] T. Sasao, "On the minimization of partially defined classification functions,' *ISMVL-2020*, Nov. 8-10, 2020, Miyazaki, Japan, pp. 117-123.

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

[22] https://en.wikipedia.org/wiki/Tic-tac-toe

[23] B. Zupan, I. Bratko, M. Bohanec, and J. Demsar, "Function decomposition in machine learning," *Machine Learning and Its Applications*, 2001, pp. 71-101.