PAPER *Special Section on Multiple-Valued Logic and VLSI Computing*

On Easily Reconstructable Logic Functions

Tsutomu SASAO[†a)], ***Member***

SUMMARY This paper shows that sum-of-product expression (SOP) minimization produces the generalization ability. We show this in three steps. First, various classes of SOPs are generated. Second, minterms of SOP are randomly selected to generate partially defined functions. And, third, from the partially defined functions, original functions are reconstructed by SOP minimization. We consider Achilles heel functions, majority functions, monotone increasing cascade functions, functions generated from random SOPs, monotone increasing random SOPs, circle functions, and globe functions. As for the generalization ability, the presented method is compared with Naive Bayes, multi-level perceptron, support vector machine, JRIP, J48, and random forest. For these functions, in many cases, only 10% of the input combinations are sufficient to reconstruct more than 90% of the truth tables of the original functions.

key words: *complexity of logic function, random function, monotone function, threshold function, logic minimization, partially defined function, classification, data mining, machine learning, generalization ability*

1. Introduction

Memorization is to memorize the training set, and can be performed by just storing the training set into a memory device. On the other hand, **learning** not only memorizes the training set, but also predicts outputs for unknown inputs that occur in the test set. Thus, learning includes the **generalization ability**. In machine learning, increasing the generalization ability is a very important issue.

In [21], we showed that a logic minimizer can be used for machine learning in handwritten digit recognition, where unknown values are predicted by assigning 0 or 1 to *don't care* values by a logic minimizer. However, it was not known if the statement is true for other classes of functions.

In this paper, we investigate how a logic minimizer predicts the unknown values for special classes of functions. We are interested in the class of functions that are easy to reconstruct. Functions that have simple representations are easy to reconstruct. Our method is as follows: Given a function f represented by a simple sum-of-products expression (SOP), randomly select the minterms of f to generate a partially defined function $\hat{f}$. From $\hat{f}$, we reconstruct the original function f using an SOP minimizer.

When the faction of the selected minterms is large, the original function can be reconstructed easily. However, when the fraction of the selected minterms is small, the reconstruction of the original function is difficult. Benchmark functions include: Achilles heel functions; symmetric threshold functions; randomly generated non-monotone SOP, randomly generated monotone SOP, circle functions, and globe functions. For these functions, we compare the performance of machine learning methods including:

- SOP minimization of partially defined function.
- Naive Bayes method.
- Multi-level perceptron (neural network).
- Support vector machine.
- Decision tree.
- Random forest.

The rest of this paper is organized as follows: Sect. 2 shows definitions of various functions and their properties. Section 3 shows the method to perform experiments. Section 4 shows the method to evaluate the performance of the reconstruction. Section 5 compares the performance of various machine learning methods. Section 6 shows the experiments with multi-valued input functions. Section 7 surveys related works. Section 8 concludes the paper.

Preliminary versions of this paper were presented in [23] and [24].

2. Definitions

Definition 2.1: Let ON, OFF, and DC be subsets of B^n, where $B = \{0, 1\}$, $ON \cap OFF = \emptyset$, $ON \cap DC = \emptyset$, $OFF \cap DC = \emptyset$, and $ON \cup OFF \cup DC = B^n$. Consider a function f such that, for any $\vec{a} \in B^n$,

$$\vec{a} \in ON \Rightarrow f(\vec{a}) = 1,$$
$$\vec{a} \in OFF \Rightarrow f(\vec{a}) = 0.$$

When $DC = \emptyset$, f is **totally defined**, while when $DC \neq \emptyset$, f is **partially defined**.

Problem 2.1: Given a partially defined function f, find the simplest sum-of-products expression (SOP) that is consistent with f.

This process is called **logic minimization**. However, it can be also used for **reconstruction of a function**.

Definition 2.2: A **minimum sum-of-products expression** (minimum SOP) for f satisfies the following conditions:

1. It has the fewest products.
2. It has the fewest literals subject to the condition 1).

Manuscript received September 27, 2023.
Manuscript revised January 29, 2024.
Manuscript publicized April 16, 2024.

[†]Department of Computer Science, Meiji University, Kawasaki-shi, 214–8571 Japan.
a) E-mail: sasao@ieee.org

DOI: 10.1587/transinf.2023LOP0001

When n is small (say $n \leq 10$), we can use the Quine-McCluskey method [13] to derive a minimum SOP. However, when n is large, exact minimization is too time-consuming. So, we have to resort to a heuristic minimization algorithm such as MINI [8] to obtain near minimum solutions.

Lemma 2.1: [20] A minimum SOP can be represented as a sum of only **prime implicants** (PIs).

Definition 2.3: The **SOP degree** of f is the maximum number of literals in a product of a minimum SOP for f across all products.

Example 2.1: Consider the partially defined function $\hat{f}$ whose ON and OFF sets are shown in Table 1. Suppose that Table 1 is the training set. Predict the output for the function for the input $\vec{a} = (x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$. Since the vector $\vec{a}$ is not contained in the training set, nobody knows the output for this input.

Figure 1 shows the map for $\hat{f}$, where blank cells denote *don't cares*. If we perform an SOP minimization using the map in Fig. 2, we have the simplified expression for $\hat{f}$: $\mathcal{F} = x_1 x_2$. Although the value of $f(\vec{a})$ is undefined, if the value is 1, and if the values of f for other blank cells are 0, then the SOP for f becomes simpler. **Occam's razor** [7] recommends to use simple rules. So, we assume that the function value for $\vec{a}$ to be 1.

Next, predict the output value for the input $\vec{b} = (0, 0, 0, 1)$. In this case, the output is assumed to be 0, since this makes the SOP simpler. Such operation is called **generalization** in machine learning. ■

Table 1 Partially defined function.

	x_1	x_2	x_3	x_4	$\hat{f}$
ON	1	1	0	0	1
	1	1	1	0	1
	1	1	1	1	1
OFF	0	0	0	0	0
	0	1	0	0	0
	0	1	0	1	0
	1	0	0	0	0
	1	0	1	0	0

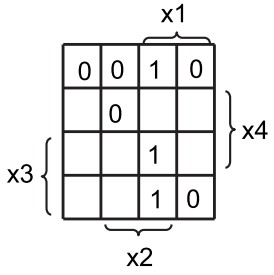

Fig. 1 Partially defined function $\hat{f}$.

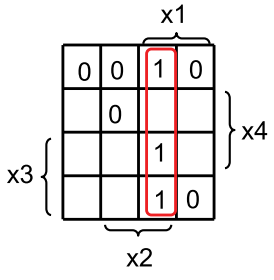

Fig. 2 Simplified SOP for the function $\hat{f}$.

Definition 2.4: $Ach(k, m)$ is the Achilles heel function of $k \times m$ variables. Each product of the minimum SOP of $Ach(k, m)$ has k literals, and the SOP has m products, where all literals are distinct.

Example 2.2:

$$
\begin{aligned}
Ach(2,6) &= x_1y_1 \vee x_2y_2 \vee x_3y_3 \vee x_4y_4 \vee x_5y_5 \vee x_6y_6 \\
Ach(3,4) &= x_1y_1z_1 \vee x_2y_2z_2 \vee x_3y_3z_3 \vee x_4y_4z_4 \\
Ach(4,3) &= x_1y_1z_1w_1 \vee x_2y_2z_2w_2 \vee x_3y_3z_3w_3 \\
Ach(6,2) &= x_1y_1z_1w_1u_1v_1 \vee x_2y_2z_2w_2u_2v_2
\end{aligned}
$$

■

Lemma 2.2: The SOP degrees of functions are

- 1 for n-variable OR function.
- n for n-variable AND, and parity function.
- $r + 1$ for $(2r + 1)$-variable majority function.
- k for Achilles heel function $Ach(k, m)$.

Example 2.3: Consider the Achilles heel function:

$$Ach(2,3) = x_1y_1 \vee x_2y_2 \vee x_3y_3.$$

The complement of Ach(2,3) is

$$
\begin{aligned}
\overline{Ach(2,3)} &= (\bar{x}_1 \vee \bar{y}_1)(\bar{x}_2 \vee \bar{y}_2)(\bar{x}_3 \vee \bar{y}_3) \\
&= \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{y}_3 \vee \bar{x}_1\bar{y}_2\bar{x}_3 \vee \bar{x}_1\bar{y}_2\bar{y}_3 \vee \\
&\quad \bar{y}_1\bar{x}_2\bar{x}_3 \vee \bar{y}_1\bar{x}_2\bar{y}_3 \vee \bar{y}_1\bar{y}_2\bar{x}_3 \vee \bar{y}_1\bar{y}_2\bar{y}_3.
\end{aligned}
$$

The last SOP is the minimum, and the maximum number of literals is three. So, the SOP degree of $\overline{Ach(2,3)}$ is three. ■

Example 2.4: Consider the partially defined function f in Table 2, where the OFF and the ON sets are shown. Note that the ON set consists of two vectors: $\{\vec{a}_1, \vec{a}_2\}$, while the OFF set consists of two vectors:$\{\vec{b}_1, \vec{b}_2\}$.

The minimal SOPs for f are $\mathcal{F}_1 = x_1x_4 \vee \bar{x}_1\bar{x}_4$, and $\mathcal{F}_2 = \bar{x}_2 \vee x_3$. The maps for these SOPs are shown in Figs. 3 and 4. The minimal SOPs for $\bar{f}$ are $\mathcal{F}_3 = x_1\bar{x}_4 \vee \bar{x}_1x_4$, and $\mathcal{F}_4 = x_2\bar{x}_3$. Thus, $\mathcal{F}_4$ is the exact minimum. The maps of these SOPs are shown in Figs. 5 and 6. The maximum number of literals in a product is two. Note that these SOPs depend on only two variables. ■

Table 2 Partially defined function.

		x_1	x_2	x_3	x_4	f
ON	$\vec{a}_1$	1	0	0	1	1
	$\vec{a}_2$	0	1	1	0	1
OFF	$\vec{b}_1$	1	1	0	0	0
	$\vec{b}_2$	0	1	0	1	0

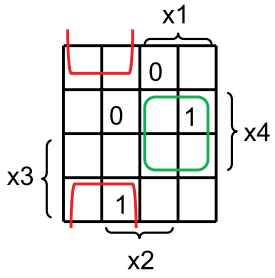

Fig. 3 SOP for f: $\mathcal{F}_1$.

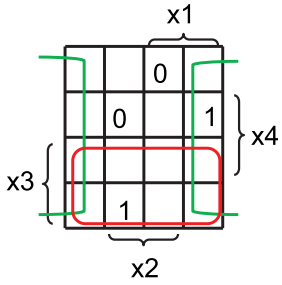

Fig. 4 SOP for f: $\mathcal{F}_2$.

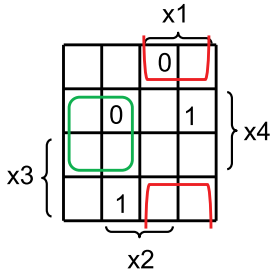

Fig. 5 SOP for $\bar{f}$: $\mathcal{F}_3$.

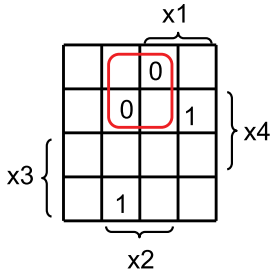

Fig. 6 SOP for $\bar{f}$: $\mathcal{F}_4$.

Definition 2.5: Let $\vec{a}$ and $\vec{b}$ be elements of B^n. If f satisfies $f(\vec{a}) \geq f(\vec{b})$, for any vectors such that $\vec{a} \geq \vec{b}$, then f is a **monotone increasing function**.

Theorem 2.1: [20] Let f be a monotone increasing function of n variables. Then,

1) All the prime implicants of f are essential prime implicants.
2) There is a unique minimum SOP for f.

3. Experiments

3.1 SOP Minimizer for Machine Learning

In this experiment, a modified version of the MINI [8], [9] logic minimizer was used. In MINI, the number of products is reduced as the primary objective, and the number of literals is reduced as the secondary objective. The following algorithm shows the outline:

Algorithm 3.1: (MINI13)

1. From the ON and the OFF sets, generate the DC set by $DC = \overline{ON \cup OFF}$.
2. Simplify the SOP for the ON set, and the SOP for the OFF set using DC, independently.
3. Count the products in the simplified SOPs. Let f_1 and f_0 be the totally defined functions for the simplified SOPs for the ON and the OFF sets, respectively.
4. If the simplified SOP for f_1 has fewer products, then replace the SOP for f_0 by the simplified SOP for $f_0\overline{f_1}$. Otherwise, replace the simplified SOP for f_1 by the simplified SOP for $f_1\overline{f_0}$.

This logic minimizer has two outputs $(\mathcal{F}_p, \mathcal{F}_n)$: $(1,0)$ shows the *positive* class (1), $(0,1)$ shows the *negative* class (0), and $(0,0)$ shows **unknown**. Thus, the minimizer has a three-valued output, and shows more information than a binary logic minimizer. When the minimizer cannot decide whether an input is positive or negative, it produces **unknown** outputs. Multiple classifiers of this type can be combined to create a multi-class classifier [21]. Let $\mathcal{F}_P$ be an SOP for f_1, and let $\mathcal{F}_N$ be an SOP for $f_0\bar{f}_1$. Note that $\mathcal{F}_\mathcal{P} \cdot \mathcal{F}_\mathcal{N}$=0, but $\mathcal{F}_\mathcal{P} \vee \mathcal{F}_\mathcal{N}$ is not necessarily a tautology: $\mathcal{F}_\mathcal{P} \vee \mathcal{F}_\mathcal{N}$ only covers a part of the entire combinations.

The last step of Algorithm 3.1 is used to make the resulting SOPs disjoint. Also, it improves the generalization ability for imbalanced data sets, which will be shown in Sect. 5.1.

In the **expand-minority strategy**[†], the SOP for the minority class is simplified as the first priority.

3.2 Achilles Heel Function

Example 3.5: Consider the following Achilles heel functions: Ach(2,6), Ach(3,4), Ach(4,3), and Ach(6,2). These functions all have 12 variables. Thus, the total numbers of input combinations are $2^{12} = 4096$. The numbers of minterms in the ON sets for Ach(2,6), Ach(3,4), Ach(4,3), and Ach(6,2) are 3367, 1695, 721, and 127, respectively.

We generated partially defined functions with different numbers of selected minterms, and tried to reconstruct the original functions by logic minimization (MINI13). The results are shown in Table 3. In the table, ◯ shows that the logic minimizer successfully reconstructed the original function,

[†]The expand operation in MINI [8] reduces the number of literals in a product.

Table 3 Reconstruction of Achilles heel functions ($n = 12$).

	# of Minterms			# of Products		SOP
	100	200	400	f	Not(f)	degree
$Ach(2,6)$	×	○	○	6	64	2
$Ach(3,4)$	○	○	○	4	81	3
$Ach(4,3)$	○	○	○	3	64	4
$Ach(6,2)$	×	×	○	2	36	6

Table 4 Reconstruction of other monotone increasing functions ($n = 8$).

	# of Minterms				# of Products		SOP
	60	120	160	200	f	Not(f)	degree
Th(8,2)	×	×	×	×	28	8	2
Th(8,4)	×	×	×	×	70	56	4
MCF(8)	×	×	○	○	5	4	5

while × represents a function that was not reconstructed. A single difference in even one minterm was deemed to not be reconstructable.

This result shows that $Ach(3,4)$ and $Ach(4,3)$ are easier to be reconstructed, while $Ach(6,2)$ is harder to be reconstructed. Also, with the increase the number of selected minterms, the reconstruction of functions became easier. ■

3.3 Other Monotone Increasing Functions

A **symmetric threshold function** with n variables and threshold T is defined as

$$Th(n,T)(x_1,x_2,\ldots,x_n) = 1 \text{ iff } \sum_{i=1}^{n} x_i \geq T.$$

It is a monotone increasing function.

Th(8,4) is an 8-variable symmetric threshold function, where the threshold is four. The SOP for Th(8,4) has $\binom{8}{4} = 70$ prime implicants, while the SOP for $\overline{\text{Th}(8,4)}$ has $\binom{8}{5} = 56$ prime implicants.

Table 4 shows the experimental results. SOP minimizations could not reconstruct Th(n,T). For these functions, reconstruction did not occur because the number of the prime implicants is too large.

A **monotone cascade function** (MCF) can be realized by a cascade of two-input logic cells, where OR cells and AND cells are connected alternately. Specifically,

$$MCF(8) = (((x_1 \vee x_2)x_3 \vee x_4)x_5 \vee x_6)x_7 \vee x_8.$$

The numbers of the prime implicants of MCF(n) is $\lfloor \frac{n}{2} \rfloor + 1$, while that of the complement of MCF(n) is $\lceil \frac{n}{2} \rceil$.

Table 4 also shows the results for an MCF function. Reconstruction of the original functions from the MCF(n) functions with randomly selected minterms was easier than from the Th(n,T) function with randomly selected minterms.

3.4 Functions Generated by Random Expressions

To specify the complexity of SOPs, we use the following parameters: n is the number of variables; m is the number of products; and k is the number of literals, in each product.

Table 5 Reconstruction of random functions ($n = 8$).

	# of Minterms				# of Products		SOP
N	80	100	120	160	f	Not(f)	degree
SOP(8,4,3)	○	○	○	○	4	11	3
SOP(8,5,3)	×	×	×	○	5	15	3
SOP(8,6,3)	×	×	○	○	6	11	3
MoSOP(8,4,3)	×	○	○	○	4	13	3
MoSOP(8,5,3)	×	○	○	○	5	17	3
MoSOP(8,6,3)	×	○	○	○	6	15	3

SOP(n,m,k) is a randomly generated non-monotone SOP with n variables, m products, and k literals in each product. We carefully selected values of n, m, and k so that SOPs represent non-constant 1 functions [11]. To select k variables out of n variables, we used the algorithm in [3] and a pseudo random number generator.

MoSOP(n,m,k) is a randomly generated monotone SOP with n variables, m products, and k positive literals in each product. Table 5 shows the experimental results for 8-variable functions.

3.5 Circle and Globe Functions

Here, we define two mathematical functions[†].

Definition 3.6: The Circle function is

$$Circle(X,Y) = \begin{cases} 1 & \text{if } X^2 + Y^2 \geq R^2 \\ 0 & \text{otherwise.} \end{cases}$$

The Globe function is

$$Globe(X,Y,Z) = \begin{cases} 1 & \text{if } X^2 + Y^2 + Z^2 \geq R^2 \\ 0 & \text{otherwise.} \end{cases}$$

Assume that X, Y, and Z are represented by n-bit binary numbers, and $R = 2^n - 1$. The Circle function has $n_t = 2n$ variables, while the Globe function has $n_t = 3n$ variables.

Example 3.6: (Circle Function)

When $n = 3$, X and Y are represent by 3-bit numbers, and $R = (1,1,1)_2 = 7$.
When $X = (1,0,1)_2 = 5$ and $Y = (1,0,0)_2 = 4$, we have

$$X^2 + Y^2 = 5^2 + 4^2 = 25 + 16 = 41 < (7^2 = 49).$$

Thus, $Circle(5,6) = 0$.
When $X = (1,1,0)_2 = 6$ and $Y = (1,1,1)_2 = 7$, we have

$$X^2 + Y^2 = 6^2 + 7^2 = 36 + 49 = 85 > (7^2 = 49).$$

Thus, $Circle(6,7) = 1$. ■

In the case of circle functions, when n is large, the number of false minterms is proportional to $\frac{\pi}{4}R^2$, while the number of true minterms is proportional to $R^2 - \frac{\pi}{4}R^2$. Thus, with the increase of n,

$$\frac{\text{the number of true minterms}}{\text{the number of false minterms}} \rightarrow \frac{4-\pi}{\pi} = \frac{4}{\pi} - 1 \approx 0.2732.$$

[†]A circuit function is considered in [4], but the definition is different.

Table 6 Confusion matrix.

	Predicted	
Actual	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Thus, the circle function is an imbalanced function.

In the case of globe function, the ratio approaches

$$\frac{6-\pi}{\pi} = \frac{6}{\pi} - 1 \approx 0.90986,$$

as n increases. So, the globe function is not so imbalanced as the circle function.

By randomly selecting the minterms of these functions, we have partially defined functions.

4. Evaluation of Methods

To evaluate classifiers, we use the **confusion matrix** shown in Table 6, where TP is the number of **true positives**, FP is the number of **false positives**, FN is the number of **false negatives**, and TN is the number of **true negatives**.

Here, we use four measures: *Accuracy, Precision, Recall* and *Matthews Correlation Coefficient* (MCC) [5]

Definition 4.7:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$
$$Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN}$$
$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)\cdot(TP+FN)\cdot(TN+FP)\cdot(TN+FN)}}$$

If the classifier reconstructed the original function perfectly, then $FN = FP = 0$, and $Accuracy = Precision = Recall = MCC = 1.00$.

When, the fraction of the positive instances is very small, the classifier that classifies all the instances to *Negative* has a very high *Accuracy*. However, both *Precision* and *Recall* are zero. Thus, the predictive power of the classifier for the positive instances is zero.

When, the fraction of the positive or negative instances is small, the data set is called **imbalanced**. To show the real performance for imbalanced data sets, MCC is used. MCC shows the quality of two-class classifications. When $FN = FP = 0$, $MCC = 1.00$, while when $TN = TP = 0$, $MCC = -1.00$. For the classifier that classifies all the instances to *Negative* ($TP = FP = 0$), or all the instances to *Positive* ($TN = FN = 0$), MCC is undefined (UD). In this case, the denominator of MCC is zero.

Example 4.7: Consider the function shown in Fig. 7. It is an Achilles heel function with 4 variables:

$$Ach(2,2) = x_1x_2 \vee x_3x_4.$$

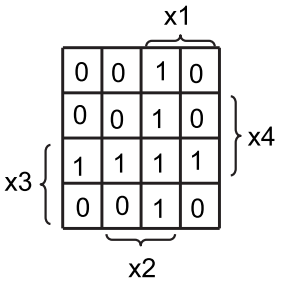

Fig. 7 Achilles heel function f.

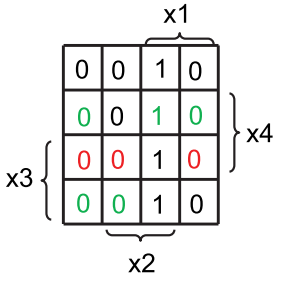

Fig. 8 Function after logic minimization of $\hat{f}$.

Consider the function shown in Fig. 1, where eight minterms are randomly selected. The blank cells are *don't cares*.

Figure 8 shows the minimized function. The colored cells are predicted by the logic minimizer. Among them, red cells are incorrectly predicted, while green cells are correctly predicted. From Fig. 8, we have the confusion matrix:

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} = \begin{bmatrix} 4 & 3 \\ 0 & 9 \end{bmatrix}$$

From this, we have Accuracy = 0.8125, and MCC = 0.6547. ■

In Sect. 3, a logic minimizer reconstructed the original functions from the functions whose minterms were randomly selected. In Tables 3, 4 and 5, instances with ○ show that the reconstructions are perfect. That is FP = FN = 0, and Accuracy = MCC = 1.00.

These experiments show that SOP minimization increased generalization ability.

5. Comparison with other Methods

5.1 Performance of the Proposed Method

Here, we analyze ten cases with × marks in Tables 3, 4 and 5. Tables 7 and 8 show experimental results. The last line headed with MINI13 in Tables 7 and 8 show the Accuracy and MCC obtained by the proposed method. In Tables 7 and 8, bold figures show the largest MCC. These data show that the logic minimizer predicted unknown data fairly well. The Accuracy is, in many cases, acceptable. However, MCC for Ach(2,6) and Ach(6,2) is lower. Note that for Ach(2,6), $|ON| = 3369$ and $|OFF| = 727$, while for Ach(6,2), $|ON| = 127$ and $|OFF| = 3969$. In other words,

Table 7 Accuracy and MCC for various classifiers (Two-valued inputs).

	Ach(2,6) 100 Minterms		Ach(6,2) 200 Minterms		Th(8,2) 200 Minterms		Th(8,4) 200 Minterms		MCF(8) 120 Minterms	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes	0.851	0.373	0.972	0.345	0.968	0.328	0.972	0.942	0.953	0.896
MLP	0.863	0.502	0.961	0.396	1.000	**1.000**	1.000	**1.000**	0.992	0.983
SMO	0.864	0.457	0.969	*UD*	0.996	0.941	1.000	**1.000**	0.961	0.918
JRIP	0.832	0.402	0.969	**0.494**	0.972	0.700	0.867	0.724	0.980	0.956
J48	0.840	0.407	0.969	*UD*	0.965	0.453	0.816	0.599	0.988	0.974
RF	0.870	0.479	0.971	0.246	0.996	0.941	0.965	0.924	0.996	**0.991**
MINI13	0.874	**0.561**	0.965	0.479	0.980	0.760	0.945	0.887	0.996	**0.991**

Table 8 Accuracy and MCC for various classifiers (Two-valued inputs).

	SOP(8,5,3) 120 Minterms		SOP(8,6,3) 100 Minterms		MoSOP(8,4,3) 80 Minterms		MoSOP(8,5,3) 80 Minterms		MoSOP(8,6,3) 80 Minterms	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
Bayes	0.738	0.476	0.730	0.441	0.809	0.575	0.812	0.615	0.801	0.597
MLP	0.770	0.544	0.883	0.759	0.832	0.638	0.859	0.707	0.879	0.755
SMO	0.730	0.463	0.746	0.479	0.797	0.546	0.844	0.674	0.840	0.674
JRIP	0.773	0.547	0.895	0.784	0.875	0.726	1.000	**1.000**	0.938	**0.874**
J48	0.766	0.535	0.832	0.679	0.859	0.689	0.812	0.609	0.906	0.813
RF	0.832	0.664	0.906	0.807	0.910	**0.804**	0.887	0.764	0.902	0.803
MINI13	0.957	**0.917**	0.984	**0.966**	0.877	0.737	0.879	0.753	0.892	0.783

Table 9 Accuracy and MCC for circle and globe functions.

Function Name	n	n_t	P	N	3/32			4/32			5/32		
					s	ACC	MCC	s	ACC	MCC	s	ACC	MCC
Circle	4	8	63	193	24	0.7305	0.2775	32	0.8320	0.5704	40	0.8468	0.5731
Circle	5	10	242	782	96	0.8467	0.6053	128	0.8908	0.7157	160	0.8630	0.6659
Circle	6	12	919	3177	384	0.9377	0.8362	512	0.9501	0.8645	640	0.9696	0.9150
Circle	7	14	3602	12782	1536	0.9702	0.9154	2048	0.9764	0.9325	2560	0.9767	0.9339
Circle	8	16	14213	51323	6144	0.9876	0.9637	8192	0.9884	0.9662	10240	0.9885	0.9665
Circle	9	18	56570	205574	24576	0.9920	0.9765	32768	0.9927	0.9786	40960	0.9935	0.9810
Globe	3	9	273	239	48	0.8870	0.7707	64	0.7404	0.5088	80	0.8750	0.7586
Globe	4	12	2074	2022	384	0.9032	0.8108	512	0.9192	0.8402	640	0.9253	0.8532
Globe	5	15	16058	16710	3072	0.9538	0.9082	4096	0.9614	0.9234	5120	0.9665	0.9335
Globe	6	18	126510	135634	24576	0.9751	0.9505	32768	0.9778	0.9557	40960	0.9808	0.9618

Table 10 Comparison of two minimizers on circle and globe functions.

Function Name	n	Minterms s	MINI13 (Expand-Minority)				MINI10 (Expand-Both)			
			ACC	MCC	t_p	t_n	ACC	MCC	t_p	t_n
Circle	4	30	0.836	**0.539**	2	9	0.798	0.402	2	3
Circle	5	100	0.819	0.570	8	33	0.839	**0.577**	7	8
Circle	6	400	0.956	**0.878**	13	71	0.932	0.812	14	19
Circle	7	1600	0.969	**0.913**	27	244	0.965	0.902	27	34
Circle	8	6000	0.986	**0.958**	60	608	0.971	0.915	60	79
Circle	9	26000	0.991	**0.975**	132	2211	0.986	0.958	129	151
Globe	3	50	0.876	**0.752**	25	6	0.819	0.642	8	6
Globe	4	400	0.886	**0.776**	20	108	0.880	0.760	22	24
Globe	5	3200	0.949	**0.899**	957	90	0.931	0.863	102	90
Globe	6	26000	0.991	**0.975**	5610	381	0.964	0.927	398	390

For the two MCC columns, boldface shows larger MCC.

these data sets are imbalanced. Experimental results show that imbalanced data sets are hard to reconstruct [12].

Table 9 shows accuracy and MCC for circle and globe functions. For example, when $n = 4$, the Circle function has $n_t = 4 \times 2 = 8$ input variables. Thus, the total number of the input combinations is $2^8 = 256$. The number of elements in the ON set is $P = 63$, while the number of elements in the OFF set is $N = 193$. Each group of three columns headed with $i/32$ shows the results when the number of minterms is $s = (i/32)2^{n_t}$ for $i = 3, 4, 5$. The table shows that with the increase of the number of minterms, the ACC and MCC tend to increase. Also, with the increase of n, ACC and MCC increase. When $n_t \geq 12$, the experimental result shows that the training set with only 10% of the total combinations are sufficient to predict more than 90% of the outputs[†].

Table 10 compares the effect of minimization algorithms. The columns headed *MINI13* show the case when

[†]In the globe function with $n = 4$, ACC and MCC for s = 410 are 0.9147 and 0.8337, respectively.

Table 11 Accuracy and MCC for random SOPs (Four-valued inputs: MINI13).

Function	100 Minterms		150 Minterms		200 Minterms		300 Minterms		400 Minterms		600 Minterms	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
MVSOP(5,3,2)	0.973	**0.933**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	1.000	1.000	1.000
MVSOP(5,4,2)	0.968	**0.918**	0.972	**0.929**	1.000	**1.000**	1.000	1.000	1.000	1.000	1.000	1.000
MVSOP(5,6,2)	0.978	**0.932**	0.951	**0.868**	0.970	0.909	0.979	0.936	0.984	0.953	1.000	**1.000**
MVSOP(5,3,3)	0.911	0.813	0.968	0.933	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MVSOP(5,4,3)	0.949	**0.902**	0.985	**0.971**	0.997	**0.994**	0.997	0.994	0.997	0.994	0.997	0.994
MVSOP(5,6,3)	0.939	**0.878**	0.873	**0.762**	0.957	0.913	0.996	**0.992**	0.990	0.980	0.992	0.984

MINI13 was used, while the columns headed *MINI10* show the case when **MINI10** was used. MINI10 is similar to MINI13, but step 4 in Algorithm 3.1 is removed. Thus, both the ON cover and the OFF are expanded, and the positive and the negative outputs can be true at the same time. This is called the **expand-both strategy**. In this case, the classifier produces four different outputs. $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$, where $(0,0)$ and $(1,1)$ show **unknown** class. Since MINI10 produced $(1,1)$ outputs, we have $TP+FP+FN+TN > 2^N$. Although, MINI10 produced solutions with smaller t_n than MINI13, it often produced solutions with lower Accuracy and MCC than MINI13 when $n_t \geq 10$.

In Table 10, bold face shows larger MCC values for the two MCC columns. MINI13 produced higher MCC for 9 functions, while MINI10 produced higher MCC for only one functions.

5.2 Performance of Other Methods

We investigated the performance of the following classifiers in the WEKA [26] system.

- *Bayes* is a statistical learning algorithm based on Bayes' theorem. It is also called Naive Bayes method. It assumes that variables are independent.
- *MLP* (a Multi-Layer Perceptron) is a feed-forward artificial neural network.
- *SMO* is an extension of a support vector machine using a sequential minimal optimization algorithm [16].
- *JRIP* is a rule learner based on the RIPPER (Repeatedly Incremental Pruning to Produce Error Reduction) algorithm [6].
- *J48* is a decision tree classifier, and is a Java implementation of C4.5 algorithm [17].
- *Random Forest* (RF) is an ensemble classifier that consists of many decision trees.

The parameters for classifiers were set to default values of WEKA. As for the test data, we used the set of all input combinations, i.e., 2^n vectors, since we know the correct values for all possible input combinations[†].

Tables 7 and 8 also compare the performance of other methods: Bayes, MLP, SMO, JRIP, J48, and RF.

- The bold numbers show the highest MCCs.
- The method that produced rules with the highest MCC also produced rules with the highest Accuracy.
- Bayes produced the lowest MCCs for five functions.
- MLP produced the highest MCCs for Th$(8,2)$ and Th$(8,4)$.
- SMO produced the highest MCCs for Th$(8,4)$. This result is reasonable, since Th$(8,4)$ can be represented by a simple threshold gate.
- JRIP produced the highest MCCs for three functions.
- RF produced the highest MCCs for two functions.
- MINI13 produced the highest MCCs for four functions.
- For Ach$(6,2)$, two algorithms SMO and J48 produced MCC with UD. In these cases, the algorithms classified all the data into the negative class. Thus, TP = FP = 0, and the values of MCC and *Precision* are undefined (UD). And, *Recall* is 0.00. This function is imbalanced, and is hard to reconstruct.

In addition to the Accuracy and MCC, we have to consider the complexity of the models (rules). In general, MLP, J48 and RF are too complex to analyze, while SOPs generated by JRIP and MINI13 are relatively easy to analyze.

6. Extension to Multi-Valued Input Functions

In this part, we extend the theory to multi-valued input functions. For simplicity, we consider the data sets for the functions that are generated by random SOPs:

$$f : \{0,1,2,3\}^5 \rightarrow \{0,1\}.$$

We assume that each SOP has m products and k literals, and each literal has one of the following forms:

$$X^{\{3\}}, X^{\{2,3\}}, X^{\{1,2,3\}}, X^{\{0,1,2,3\}}.$$

This implies that the functions are monotone increasing. UCI benchmark functions [25] such as *Breast Cancer Wisconsin (Original)*, *Car Evaluation*, and *Nursery* have such a property.

We did similar experiments to the two-valued cases. Table 11 shows the experimental results. The first column shows the function name: MVSOP(n,m,k), where n denotes the number of variables, m denotes the number of products, and k denotes the number of literals in each product. Note that the total numbers of input combinations for these functions are $4^5 = 1024$. The first row of Table 11 shows that to reconstruct $MVSOP(5,3,2)$, 100 selected minterms were not sufficient, but with 150 selected minterms, MINI13 could reconstruct the original function. With the increase of the

[†]This is different from a common method to measure the accuracy, since generation of all possible input combinations are usually impractical.

Table 12 Accuracy and MCC for random SOPs (Four-valued inputs: JRIP).

Function	100 Minterms		150 Minterms		200 Minterms		300 Minterms		400 Minterms		600 Minterms	
	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC	ACC	MCC
MVSOP(5,3,2)	0.941	0.873	0.973	0.933	0.988	0.971	0.988	0.971	1.000	1.000	1.000	1.000
MVSOP(5,4,2)	0.910	0.787	0.957	0.893	0.988	0.972	1.000	1.000	1.000	1.000	1.000	1.000
MVSOP(5,6,2)	0.885	0.610	0.947	0.846	0.990	**0.970**	0.990	**0.970**	0.994	**0.983**	0.998	0.994
MVSOP(5,3,3)	0.953	**0.900**	1.000	**1.000**	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
MVSOP(5,4,3)	0.921	0.850	0.974	0.948	0.996	0.992	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**
MVSOP(5,6,3)	0.889	0.783	0.936	0.871	0.959	**0.918**	0.957	0.914	0.992	**0.984**	0.996	**0.992**

selected minterms, the *Accuracy* and *MCC* tend to increase. Also, with the increase of the numbers of products (m) and literals (k), the reconstruction tend to be more difficult.

When the numbers of products m and the number of literals k in each product are small, less than 10% of the input combinations are sufficient to reconstruct more than 90% of the truth table of the original functions. Thus, SOP minimizations improved the generalization ability in the case of multi-valued inputs.

To compare with an existing method, we did a similar experiment using the JRIP program, which is a rule-based method for machine learning [6]. Table 12 shows the results. In Tables 11 and 12, boldface shows larger MCC values across MINI13 (Table 11) and JRIP (Table 12). Table 11 has 17 boldface entries, while Table 12 has 11 boldface entries. Thus, roughly speaking, MINI13 performed better than JRIP.

7. Related Works

Learning of Boolean functions was considered in the papers [1], [2], and [15]. Especially, Aizenstein and Pitt [1] considered the number of selected minterms to reconstruct the original SOPs.

Muselli and Ferrari [14] used SOPs with limited degree as a model for machine learning. They considered the reconstruction of partially defined monotone increasing logic functions, and analyzed computational complexity of the learning algorithm. They analyzed the complexity of computing. Hong [9] showed that a logic minimizer for circuit design [8] can be used for machine learning. The results of IWLS contest for reconstruction of logic functions are summarized in [18]. Learning of logic functions using support vector machines was considered in the paper [19]. It compared its performance with C4.5 and Naive Bayes classifiers. Experimental results of a rule-based classifier for various benchmark functions were also shown in the paper [10]. Sasao et al. [21] designed an MNIST character recognition circuit using LUTs. They used a ternary output SOP minimizer to improve generalization ability for multi-class problems.

8. Conclusion

In this paper, we showed that SOP minimization produces the generalization ability for various classes of functions. Experimental results showed that the following functions are easily reconstructable:

- Functions with a small number of products in their SOPs.
- Functions with a small number of literals in each products in their SOPs.
- Monotone functions.
- Circuit and globe functions.

Especially, in circle and globe functions with $n_t \geq 12$, only 10% of the input combinations are sufficient to reconstruct more than 90% of the truth tables of the original functions.

Also, we compared the expand-minority strategy in MINI13 algorithm, and the expand-both strategy in MINI10 algorithm, and found that the expand-minority strategy often produces better Accuracy and MCC than the expand-both strategy.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. Prof. Jon T. Butler and reviewers' comments improved presentation.

References

[1] H. Aizenstein and L. Pitt, "On the learnability of disjunctive normal form formulas," Machine Learning, vol.19, no.3, pp.183–208, June 1995.
[2] A. Blum, C. Burcht, and J. Langford, "On learning monotone boolean functions," Proc. Symposium on Foundations of Computer Science, FOCS-1998, pp.408–415, 1998.
[3] J.T. Butler and T. Sasao, "Index to constant weight codeword converter," 7th International Symposium on Applied Reconfigurable Computing (ARC 2011), Lecture Notes in Computer Science, vol.6578, pp.193–205, 2011.
[4] S. Chatterjee, "Learning and memorization," International Conference on Machine Learning (ICML 2018), pp.754–762, 2018,
[5] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," BMC Genomics, vol.21, no.6, 2020.
[6] W.W. Cohen, "Fast effective rule induction," Twelfth International Conference on Machine Learning, pp.115–123, 1995.
[7] P. Domingos, "The role of Occam's razor in knowledge discovery," Data Mining and Knowledge Discovery, vol.3, pp.409–425, 1999.
[8] 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.
[9] 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.
[10] M.H. Ibrahim and M. Hacibeyoglu, "A novel switching function approach for data mining classification problems," Soft Comput,

vol.24, pp.4941–4957, 2020.
[11] S. Kirkpatrick and B. Selman, "Critical behavior in the satisfiability of random boolean expressions," Science, vol.264, no.5163, pp.1297–1301, 1994.
[12] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," Progress in Artificial Intelligence, vol.5, pp.221–232, 2016.
[13] S. Muroga, Logic Design and Switching Theory, Wiley-Interscience Publication, 1979.
[14] M. Muselli and E. Ferrari, "Coupling logical analysis of data and shadow clustering for partially defined positive boolean function reconstruction," IEEE Trans. Knowl. Data Eng., vol.23, no.1, pp.37–50, Jan. 2011.
[15] B.K. Natarajan, "On learning boolean functions," Proc. ACM Symposium on Theory of Computing (STOC-1987), pp.296–304, Jan. 1987.
[16] J.C. Platt, "Fast training of support vector machines using sequential minimal optimization," C.J.C Burges, A.J. Smola, and B. Schölkopf, eds., Advances in Kernel Methods — Support Vector Learning, MIT Press, Jan. 1998.
[17] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, California, 1993.
[18] S. Rai, W.L. Neto, Y. Miyasaka, X. Zhang, M. Yu, Q. Yi, M. Fujita, G.B. Manske, M.F. Pontes, L.S. da Rosa, M.S. de Aguiar, P.F. Butzen, P.-C. Chien, Y.-S. Huang, H.-R. Wang, J.-H.R. Jiang, J. Gu, Z. Zhao, Z. Jiang, D.Z. Pan, B.A. de Abreu, I. de Souza Campos, A. Berndt, C. Meinhardt, J.T. Carvalho, M. Grellert, S. Bampi, A. Lohana, A. Kumar, W. Zeng, A. Davoodi, R.O. Topaloglu, Y. Zhou, J. Dotzel, Y. Zhang, H. Wang, Z. Zhang, V. Tenace, P.-E. Gaillardon, A. Mishchenko, and S. Chatterjee, "Logic synthesis meets machine learning: Trading exactness for generalization," DATE2021, pp.1026–1031, 2021.
[19] K. Sadohara, "On a capacity control using boolean kernels for the learning of boolean functions," 2002 IEEE International Conference on Data Mining, pp.410–417, 2002.
[20] T. Sasao, Switching Theory for Logic Synthesis, Springer, 1999.
[21] T. Sasao, Y. Horikawa, and Y. Iguchi, "Classification functions for handwritten digit recognition," IEICE Trans. Inf. & Syst., vol.E104-D. no.8, pp.1076–1082, Aug. 2021.
[22] T. Sasao, "A method to generate rules from examples," International Symposium on Multiple-Valued Logic (ISMVL-2022), pp.176–181, May 2022.
[23] T. Sasao, "Easily reconstructable logic functions," International Symposium on Multiple-Valued Logic (ISMVL-2023), pp.12–17, 2023.
[24] T. Sasao, Classification Functions for Machine Learning and Data Mining, Springer Nature, Aug. 2023.
[25] https://archive.ics.uci.edu/ml/datasets.php (Accessed 1 June 2023)
[26] https://www.cs.waikato.ac.nz/ml/weka/index.html (Accessed 2 Feb. 2023)

Tsutomu Sasao received the B.E., M.E., and Ph.D. degrees in Electronics Engineering from Osaka University, Osaka Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan; IBM T.J. Watson Research Center, Yorktown Height, NY; the Naval Postgraduate School, Monterey, CA; and Kyushu Institute of Technology, Iizuka, Japan; and Meiji University, Kawasaki, Japan. Now, he is a visiting researcher at Meiji University, Kawasaki, Japan. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published more than nine books on logic design including, Logic Synthesis and Optimization, Representation of Discrete Functions, Switching Theory for Logic Synthesis, Logic Synthesis and Verification, Memory-Based Logic Synthesis, Index Generation Functions, and Classification Functions for Machine Learning and Data Mining, in 1993, 1996, 1999, 2001, 2011, 2019, and 2023 respectively. He has served Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan in 1998. He received the NIWA Memorial Award in 1979, Takeda Techno-Entrepreneurship Award in 2001, and Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVLs in 1986, 1996, 2003, 2004 and 2012. He has served as an associate editor of the IEEE Transactions on Computers. He is a Life Fellow of the IEEE.