Easily Reconstructable Logic Functions

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

***Abstract*—This paper shows that sum-of-product expression (SOP) minimization produces the generalization ability. We show this in three steps. First, various classes 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, and monotone increasing random SOPs. As for generalization ability, the presented method is compared with Naive Bayes, multi-level perceptron, support vector machine, JRIP, J48, and random forest.**

***Index Terms*—complexity of logic function, random function, monotone function, threshold function, logic minimization, partially defined function, classification, data mining, machine learning, generalization ability.**

I. 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 in the test set. Thus, learning gains **generalization ability**. In machine learning, increasing the generalization ability is a very important issue.

In [15], we showed that a logic minimizer can be used for machine learning in handwritten digit recognition, where unknown values are predicted by *don't care* assignment of a logic minimizer.

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, and randomly generated monotone SOP. 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: Section II shows definitions of various functions and their properties. Section III shows the method to perform experiments. Section IV shows the method to evaluate the performance of the reconstruction. Section V compares the performance various machine learning methods. Section VI shows the experiments with multi-valued input functions. Section VII surveys related works. Section VIII concludes the paper.

II. Definitions

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

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

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

Problem 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 has the fewest products, and the number of literals are also minimal among all the SOPs for f.

Lemma 2.1: [14] A minimum SOP can be represented as a sum of **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 2.1. Suppose that Table 2.1 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.

Fig. 2.1 show the map for $\hat{f}$, where blank cells denote *don't cares*. If we perform an SOP minimization using the map in Fig. 2.2, we have the simplified expression for $\hat{f}$:$f = x_1 x_2$. 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

TABLE 2.1
PARTIALLY DEFINED FUNCTION

	x_1	x_2	x_3	x_4	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

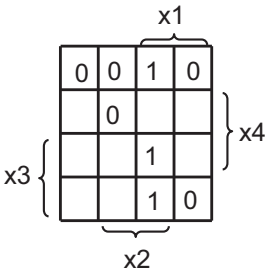

Fig. 2.1. Partially defined function f

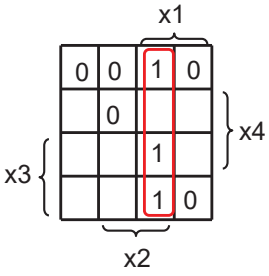

Fig. 2.2. Simplified SOP for the function $\hat{f}$.

f becomes simpler. **Occam's razor** [4] 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. ■

Lemma 2.2: The SOP degrees of functions are

- n for n-variable AND, OR, and parity function.
- $r + 1$ for $(2r + 1)$-variable majority function.
- 2 for $(2r)$-variable Achilles heel function [14]:

$$Ach2(r) = x_1y_1 \vee x_2y_2 \vee \cdots \vee x_ry_r.$$

Example 2.2: Consider the Achilles heel function with 6 variables:

$$Ach2(3) = x_1y_1 \vee x_2y_2 \vee x_3y_3.$$

The complement of Ach2(3) is

$$\begin{aligned}\overline{Ach2(3)} &= (\bar{x}_1 \vee \bar{y}_1)(\bar{x}_2 \vee \bar{y}_2)(\bar{x}_3 \vee \bar{y}_3) \\ &= \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2\bar{y}_3 \vee \bar{x}_1\bar{y}_2\bar{x}_3 \vee \bar{x}_1\bar{y}_2\bar{y}_3 \vee \\ &\quad \bar{y}_1\bar{x}_2\bar{x}_3 \vee \bar{y}_1\bar{x}_2\bar{y}_3 \vee \bar{y}_1\bar{y}_2\bar{x}_3 \vee \bar{y}_1\bar{y}_2\bar{y}_3.\end{aligned}$$

The last SOP is the minimum, and the maximum number of literal is three. So, the SOP degree of $\overline{Ach2(3)}$ is three. ■

Example 2.3: Consider the partially defined function f in Table 2.2, 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 2.3 and 2.4. 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

TABLE 2.2
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

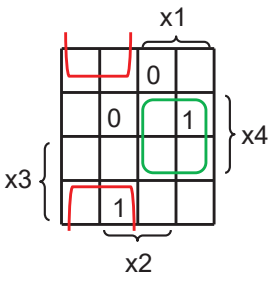

Fig. 2.3. SOP for f: $\mathcal{F}_1$.

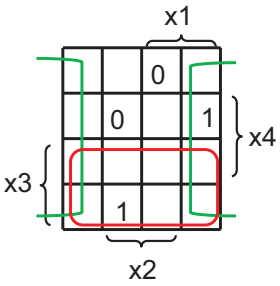

Fig. 2.4. SOP for f: $\mathcal{F}_2$.

these SOPs are shown in Figs 2.5 and 2.6. The maximum number of literals in a product is two. Note that these SOPs depend on only two variables. ■

Definition 2.4: 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: 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.

III. EXPERIMENTS

A. SOP Minimizer for Machine Learning

In this experiment, a modified version of MINI [5] [6] logic minimizer was used. The following algorithm shows the outline:

Algorithm 3.1: (MINI13)

1) From the ON and the OFF sets, generate the DC set by $DC = \overline{ON \cup OFF}$.

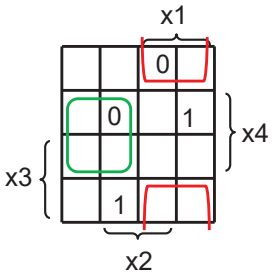

Fig. 2.5. SOP for $\bar{f}$: $\mathcal{F}_3$.

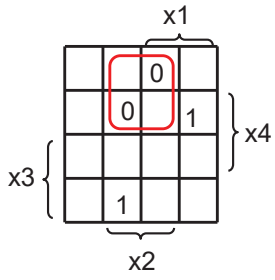

Fig. 2.6. SOP for $\bar{f}$: $\mathcal{F}_4$.

TABLE 3.1
RECONSTRUCTION OF ACHILLES HEEL FUNCTIONS ($n = 12$)

	# of Minterms			# of Products		SOP
	100	200	400	f	Not(f)	degree
$Ach2(6)$	×	◯	◯	6	64	2
$Ach3(4)$	◯	◯	◯	4	81	3
$Ach4(3)$	◯	◯	◯	3	64	4
$Ach6(2)$	×	×	◯	2	36	6

TABLE 3.2
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

TABLE 3.3
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

2) Simplify the SOP for the ON set, and the SOP for the OFF set using DC, independently.
3) Count the numbers of 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 a 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 $f_1\overline{f_0}$.

The last step is used to make the resulting SOPs disjoint. Also, it improves the generalization ability for imbalanced data set.

B. Achilles Heel Function

Example 3.1: Consider the following Achilles heel functions:

$$\begin{aligned} Ach2(6) &= x_1y_1 \vee x_2y_2 \vee x_3y_3 \vee x_4y_4 \vee x_5y_5 \vee x_6y_6. \\ Ach3(4) &= x_1y_1z_1 \vee x_2y_2z_2 \vee x_3y_3z_3 \vee x_4y_4z_4. \\ Ach4(3) &= x_1y_1z_1w_1 \vee x_2y_2z_2w_2 \vee x_3y_3z_3w_4. \\ Ach6(2) &= x_1y_1z_1w_1u_1v_1 \vee x_2y_2z_2w_2u_2v_2. \end{aligned}$$

These functions have 12 variables. Thus, the total number of input combinations is $2^{12} = 4096$. The numbers of minterms in the ON sets for Ach2(6), Ach3(4), Ach4(3), and Ach6(2) are 3367, 1695, 721, and 127, respectively.

We generated partially defined functions with different number of selected minterms, and tried to reconstruct the original functions by the logic minimizer. The results are shown in Table 3.1. In the table, ◯ shows that the logic minimizer successfully reconstructed the original function, while × shows that the logic minimizer failed to reconstruct the original function. This result shows that $Ach3(4)$ and $Ach4(3)$ are easier to be reconstructed, while $Ach6(2)$ is harder to be reconstructed. Also, with the increase the number of selected minterms, the reconstruction of functions became easier. ■

C. Other Monotone Increasing Functions

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

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

It is a monotone increasing function.

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

Table 3.2 shows the experimental results. SOP minimizations could not reconstruct Th(n,T). For these functions, the number of the prime implicants is 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.

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

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 3.2 also shows the results. 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.

D. 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.

SOP(n, m, k) is a randomly generated non-monotone SOP with n variables, m products, and k literals in each product.

MoSOP(n, m, k) is a randomly generated monotone SOP with n variables, m products, and k positive literals in each product. Table 3.3 compares the experimental results for 8-variable functions.

IV. EVALUATION OF METHODS

To evaluate classifiers, we use the **confusion matrix** shown in Table 4.1, where TP denotes **true positive**, FP denotes **false positive**, FN denotes **false negative**, and TN denotes **true negative**.

TABLE 4.1
CONFUSION MATRIX

Actual	Predicted	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Here, we use four measures: *Accuracy, Precision, Recall* and *Matthews Correlation Coefficient* (MCC) [2]

Definition 4.1:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

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

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 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 4.1: Consider the function shown in Fig. 4.1. It is an Achilles heel function with 4 variables:

$$Ach2(2) = x_1x_2 \vee x_3x_4.$$

Consider the function shown in Fig. 2.1, where eight minterms were randomly selected. The blank cells are *don't cares*.

Fig. 4.2 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. 4.2, 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 section III, a logic minimizer reconstructed the original functions from the functions whose minterms were randomly selected. In Tables 3.1, 3.2 and 3.3, instances with ◯ show that the reconstructions are perfect. That is FP=FN=0, and Accuracy=MCC=1.00.

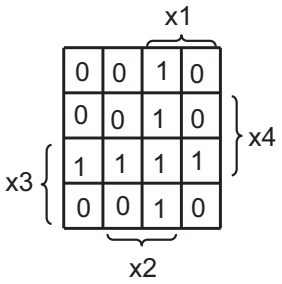

Fig. 4.1. Achilles hill function f

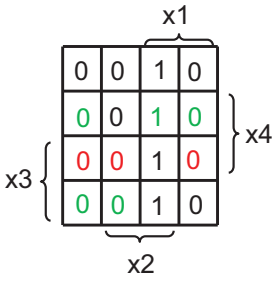

Fig. 4.2. Function after logic minimization of $\hat{f}$.

These experiments show that SOP minimization increased **generalization ability**.

V. COMPARISON WITH OTHER METHODS

A. Performance of the Proposed Method

Here, we analyze ten cases with $\times$ marks in Tables 3.1, 3.2 and 3.3. The last line headed with MINI13 in Tables 4.2 and 4.3 show the Accuracy and MCC obtained by the proposed method. In Tables 4.2 and 4.3, bold figures show the largest MCC. These data shows that the logic minimizer predicted unknown data fairly well. The Accuracy is, in many cases, acceptable. However, MCC for Ach2(6) and Ach6(2) is lower. Note that for Ach2(6), $|ON| = 3369$ and $|OFF| = 727$, while for Ach6(2), $|ON| = 127$ and $|OFF| = 3969$. In other words, these data sets are imbalanced. Imbalanced data sets are known to be hard to reconstruct [8].

B. Performance of Other Methods

We investigated the performance of the following classifiers in WEKA [18] system.

- *Bayes* is a statistical learning algorithm based on Bayes' theorem. It is also called Naive Bayes method, and 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 [11].
- *JRIP* is a rule learner based on the RIPPER (Repeatedly Incremental Pruning to Produce Error Reduction) algorithm [3].
- *J48* is a decision tree classifier, and is a Java implementation of C4.5 algorithm [12].
- *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[1].

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

TABLE 4.2
ACCURACY AND MCC FOR VARIOUS CLASSIFIERS (TWO-VALUED INPUTS)

	Ach2(6) 100 Minterms		Ach6(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 4.3
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

Tables 4.2 and 4.3 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 Ach6(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.

VI. 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:

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

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.

We did similar experiments to the two-valued cases. Table 5.1 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 a product. Note that the total numbers of input combinations for these functions are $4^5 = 1024$. The first row of Table 5.1 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 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 similar experiment using JRIP program, which is a rule-based method for machine learning [3]. Table 5.2 shows the results. Roughly speaking, MINI13 produced competitive solutions to JRIP.

VII. RELATED WORKS

Learning of Boolean functions was considered in the papers [1], [10]. Reconstruction of partially defined monotone

TABLE 5.1
ACCURACY AND MCC FOR RANDOM SOPS (FOUR-VALUED INPUTS:MINI13)

	100 Minterms		150 Minterms		200 Minterms		300 Minterms		400 Minterms		600 Minterms	
Function	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

TABLE 5.2
ACCURACY AND MCC FOR RANDOM SOPS (FOUR-VALUED INPUTS:JRIP)

	100 Minterms		150 Minterms		200 Minterms		300 Minterms		400 Minterms		600 Minterms	
Function	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**

increasing logic functions was considered in the work [9]. It analyzed the complexity of computing. Learning of logic functions using support vector machines was considered in the paper [13]. 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 [7].

VIII. CONCLUSION

In this paper, we showed that SOP minimization produces the generalization ability. 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.

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] 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.

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

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

[4] P. Domingos, "The role of Occam's razor in knowledge discovery," *Data Mining and Knowledge Discovery,* Vol. 3, pp. 409-425, 1999.

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

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

[7] M. H. Ibrahim, and M. Hacibeyoglu, "A novel switching function approach for data mining classification problems," *Soft Comput*, vol.24, pp.4941-4957, 2020.

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

[9] M. Muselli and E. Ferrari, "Coupling logical analysis of data and shadow clustering for partially defined positive Boolean function reconstruction," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 1, pp. 37-50, Jan. 2011.

[10] B. K. Natarajan,"On learning Boolean functions," *Proc.ACM Symposium on Theory of Computing*, (STOC-1987), pp. 296-304, Jan. 1987.

[11] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," In B. Schoelkopf and C. Burges and A. Smola, ed., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Jan. 1998.

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

[13] 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.

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

[15] T. Sasao, Y. Horikawa, and Y. Iguchi, "Classification functions for handwritten digit recognition," *IEICE Transactions on Information and Systems*, Vol. 104. No. 8 , pp.1076-1082, Aug. 2021.

[16] T. Sasao, "A method to generate classification rules from examples," *International Symposium on Multiple-Valued Logic*, (ISMVL-2022), May 2022.

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

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