J. of Mult.-Valued Logic & Soft Computing, Vol. 45, pp. 133–151
Reprints available directly from the publisher
Photocopying permitted by license only

Classification Functions for Machine Learning and Data Mining: Recent Results

TSUTOMU SASAO*

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

Accepted: October 20, 2024.

A classification function is a function that assigns integer values to inputs based on a training set. For instance, a training set consisting of images of airplanes and automobiles can be used to infer a function that distinguishes between unseen images of airplanes and automobiles. While neural networks are commonly used for this purpose, they require significant hardware resources and consume high power. This paper presents recent advancements in classification functions that address these issues. We use logic synthesis techniques to reduce the number of variables and simplify sum-of-products expressions (SOPs). This simplification improves training and hardware efficiency and reduces power dissipation, albeit at the cost of lower accuracy compared to neural networks. Experimental results for MNIST handwritten character recognition circuits and fashion-MNIST recognition circuits are presented to illustrate these techniques.

1 INTRODUCTION

Data mining is a process of discovering knowledge in sample data sets. Compact representation of large data enables us to find knowledge in the data set [3]. In many cases, the size of the sample data set is much smaller than the all possible input combinations. In such a case, the number of variables can be often reduced. By reducing the number of variables, and by simplifying the multiple-valued expression, we can derive a compact set of rules. This paper summarizes recent results on classification functions.

* Contact author: E-mail: sasao@ieee.org

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

TABLE 1
Example of Classification Function

The rest of the paper is organized as follows: Section 2 shows the definition of classification functions; Section 3 shows a method to reduce the number of variables; Section 4 shows a method to reduce the number of variables by linear decomposition; Section 5 shows relation to machine learning; Section 6 use MNIST handwritten digit recognition data sets as examples; Section 7 shows the experimental results for fashion MNIST data sets; Section 8 compares the presented method with neural networks; Section 9 introduces the PLA measure for classification functions; and Section 10 concludes the paper.

2 DEFINITIONS AND BASIC PROPERTIES

Definition 2.1. *A* **classification function** *is a mapping:*
$f : D \to \{0, 1, 2, \ldots, m-1\}$, *where* $D \subset B^n$, $B = \{0, 1\}$.
It is partially defined. m *is the number of classes. When* $m = 2$, f *is a* **two-class function**.

Example 2.1. *Table 1 shows a classification function with* $n = 6$, $m = 2$, *and* $|D| = 8$. *There are* 2^6 *possible input combinations, but the function is defined for only 8 combinations. For the other* $64 - 8 = 56$ *combinations, function values are undefined. The vectors in Table 1 are called* **registered vectors**. ■

3 REDUCTION OF VARIABLES

In this part, we show a method to reduce the number of variables in classification functions.

Example 3.1. *The function* f *in Table 1 can be uniquely represented by* x_1, x_2, x_3 *and* x_4. *That is, these four variables are sufficient to represent the*

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

TABLE 2
f can be represented with x_1, x_2, x_3 and x_4.

x_1	x_2	x_3	x_4	x_5	x_6	f	
1	1	0	0	1	1	0	*
0	1	1	0	1	1	0	
0	1	0	1	0	0	0	
0	0	0	0	1	0	0	
1	1	0	1	1	1	1	
1	0	1	1	1	1	1	
1	0	0	0	1	1	1	*
0	0	1	0	1	0	1	

TABLE 3
f cannot be represented with x_3, x_4, x_5 and x_6.

function, as shown in Table 2. All the bit patterns of the first four variables are distinct.

However, the same function cannot be represented with x_3, x_4, x_5 and x_6. Note that when $(x_3, x_4, x_5, x_6) = (0, 0, 1, 1)$, the value of f is not unique. In Table 3, the rows with asterisk show inconsistency. ■

By reducing the number of variables, we have the following advantages:

- For hardware implementation, the amount of memory can be reduced.
- For software implementation, the computation time can be reduced.
- For machine learning applications, generalization ability can be increased (Details are explained in Section 5).

Problem 1. *Develop an efficient method to minimize the number of variables for partially defined functions.*

We developed both exact and heuristic methods to minimize the number of variables [14].

Problem 2. *Derive the number of variables necessary to represent a function.*

As for randomly generated two-class functions, we have the following:

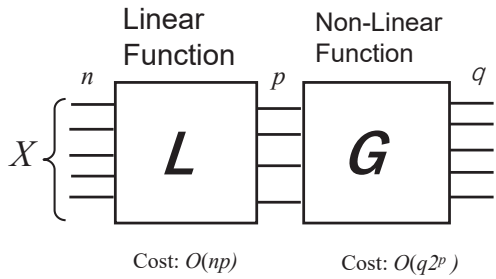

FIGURE 1
Linear decomposition.

Theorem 3.1. *[19] For randomly generated two-class functions with k_0 false minterms and k_1 true minterms, the number of variables can be reduced to $\lceil \log_2(k_0 k_1) \rceil - 2$, in most cases, where* **true minterms** *are mapped to 1, while the* **false minterms** *are mapped to 0.*

However, there exist functions where no variable can be eliminated.

4 LINEAR DECOMPOSITION

Fig. 1 shows a linear decomposition of a classification function [12, 13]. L realizes linear functions, while G realizes non-linear functions. The cost of L is $O(np)$, while the cost of G is $O(q2^p)$. We assume that L is implemented by EXOR gates, while G is implemented by a memory. When n is large, the cost of L can be neglected. The circuit L in Fig. 1 produces functions having the form:

$$y_i = a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n,$$

where $a_j \in \{0, 1\}$. y_i is called a **compound variable**. When only one element a_i is 1, and others are 0's, y_i is called a **primitive variable**. As for the number of compound variables, we have the following:

Theorem 4.1. *[15, 16, 23] With a linear decomposition, any classification function can be represented with at most $p = \lfloor \log(N_1 + 1) \rfloor$ compound variables, where*

$$N_1 = \sum_{(i<j)} k_i k_j,$$

$i, j \in \{0, 1, \ldots, m-1\}$, and k_i is the number of registered vectors $\vec{a}$ such that $f(\vec{a}) = i$.

Note that in Theorem 1, p is independent of n, the number of variables in the original function, and depends only on the numbers of registered vectors. Theorem 1 is useful for machine learning, since in most cases, the size of the training set is much smaller than 2^n, the size of the possible input combinations.

Corollary 4.1. *With a linear decomposition, any two-class function can be represented with at most $p = \lfloor \log_2(k_0 k_1 + 1) \rfloor$ compound variables, where k_0 and k_1 are the number of false and true minterms, respectively.*

Efficient algorithms to find linear decompositions [14] [23], [24] have been developed.

Example 4.1. *For the function f shown in Table 1, f can be represented as $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$. In this particular case, the general function in Fig. 1 is an identity function. That is $p = 1$, and G is just a connection.* ■

5 RELATION TO MACHINE LEARNING

Definition 5.1. *The* **generalization ability** *is the ability to predict the outcome of* **unseen data**.

As for generalization ability, we have the following [21, 27]:

- Reduction of variables improves generalization ability.
- Simplification of a sum-of-products expression (SOP) improves the generalization ability.

Example 5.1. *Consider the set of elements shown in Fig. 2. Black circles show true minterms, while black triangles show false minterms. We do not know the values of the function for the red minterms. We want to predict the values of the function for these red minterms.*

By the **nearest neighbor method** *[2], we can conjecture as shown in Fig. 3:*

- *the value for the red circle is 1, since the red circle is near to the black circles, which show true minterms.*
- *the value for the red triangle is 0, since the red triangle is near to the black triangles, which show false minterms.* ■

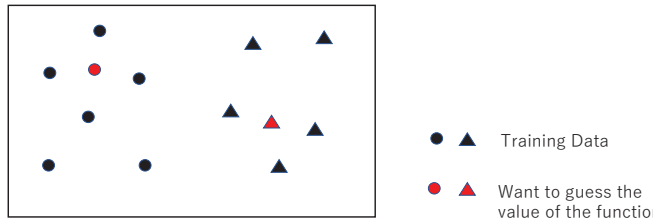

FIGURE 2
Prediction of the values for unseen data

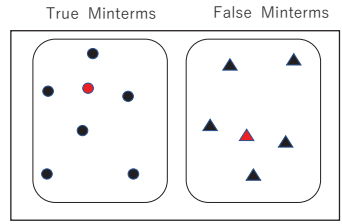

FIGURE 3
Use the value of the nearest minterms

Example 5.2. *Consider the 4-variable function shown in Fig. 4. For the blank cells and cells with question marks, we do not know the function values. We want to predict the function values for the minterms with the question marks. If we guess randomly, the probability to get the correct solution is one half for each minterm. However, by analyzing the function carefully, we can predict the correct solution with higher probability.*

After careful analysis of the function, we notice that the function is independent of the variable x_3. It can be verified that the map is symmetric with respect to the red horizontal line, as shown in Fig. 5. When the function is independent of x_3, the values for the blue cells can be guessed. In this way, the values for the minterms with question marks can be guessed. ■

Example 5.3. *Consider the 4-variable function shown in Fig. 6. For the blank cells, we do not know the function values. We want to predict the function values for the minterms with the question marks. In this case, the function depends on all the variables. Thus, we cannot use the previous approach. So, we use logic minimization.*

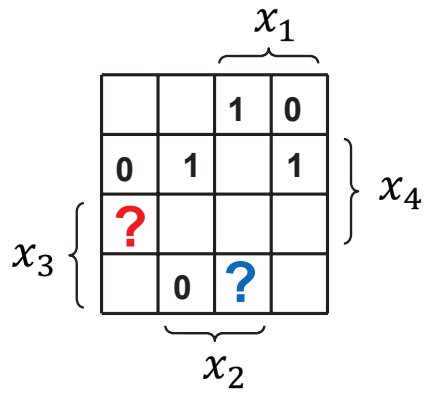

FIGURE 4
Prediction of the values for minterms with ? marks

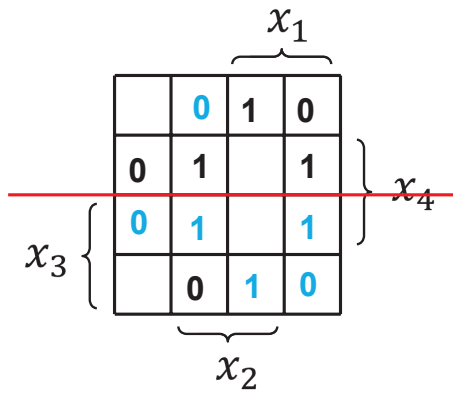

FIGURE 5
Prediction of values by variable reduction

Fig. 7 shows the minimized SOP for the function. All the true minterms are covered by three loops. From this, we can conjecture the function values for the minterms with question marks. ■

In Examples 5.2 and 5.3, we used **Occam's razor** [5] to predict the unknown values. That is, we assume that the given data set can be represented with the simplest rule.

In machine learning, known data corresponds to the **training set**, while unknown (or unseen) data correspond to the **test set**. The **training accuracy**

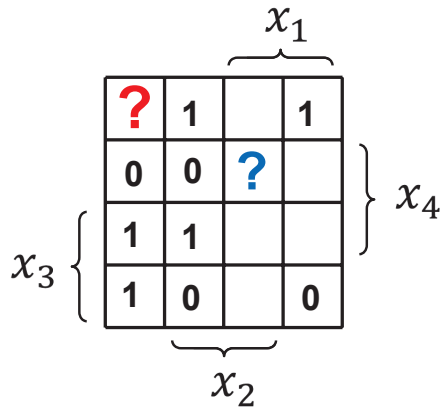

FIGURE 6
Prediction of the values for minterms with ? marks

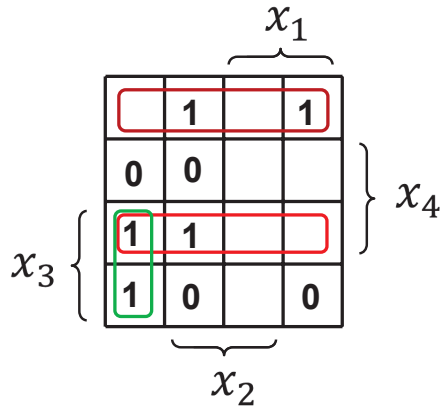

FIGURE 7
Prediction of values by SOP minimization

is the ratio of the number of correctly predicted training samples to the size of the training set. The **test accuracy** is defined similarly.

6 MNIST

MNIST [9] is a data set with handwritten digits. Fig. 8 shows selected images from the data. The data set consists of 60,000 training images, and 10,000 test images. Each image consists of $28 \times 28 = 784$ pixels. Each pixel is

FIGURE 8
Selected images of the MNIST data set.

represented by 8 bits. To reduce the data size, the original data was binarized by the following transform: **0**=[0,96] and **1**=[97, 255]. That is, pixels whose values are less than 97 are mapped to zero, while other pixels are mapped to one. In this way, the circuit f to realize this has $m = 10$ outputs (one for each digit) and $n = 784$ variables.

6.1 Result of Linear Decomposition

Consider the MNIST data set. The original function has $n = 784$ inputs, $q = 10$ outputs, and 59984 minterms. By an algorithm [14], we found a linear decomposition with $p = 37$ primitive variables in 110 seconds, and a linear decomposition with $p = 25$ compound variables in 2718 seconds* .

Note that for such a large problem, the conventional functional decomposition having the form $f(X_1, X_2) = g(h(X_1), X_2)$ [1,4] takes too much time, since the number of partitions (X_1, X_2) to consider is $O(2^n)$.

6.2 Single-Unit Realization

The single-unit realization is shown in Fig. 9, where $n = 784$ and $p = 25$. This is consistent with the upper bound on the number of compound variables:

$$UP = \lfloor \log_2(N_1 + 1) \rfloor = 30$$

* Used a computer with an INTEL Core i7, 7700, 3.65 GHz, and 64GB main memory on Windows 10.

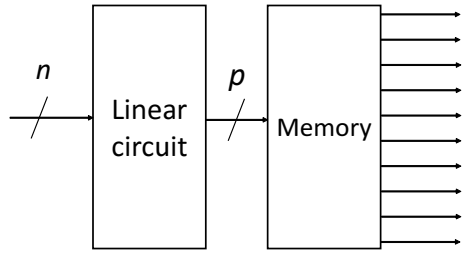

FIGURE 9
Single-unit realization of MNIST.

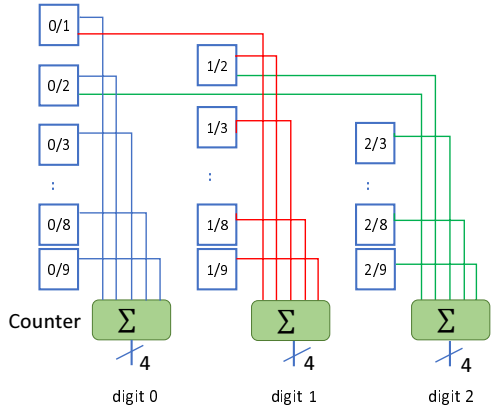

FIGURE 10
45-unit realization of MNIST.

given by Theorem 1, where

$$N_1 = \sum_{(i<j)} k_i k_j \simeq 45 \times 6000 \times 6000.$$

The linear circuit consists of 37 EXOR gates with 2 inputs. The size of the memory is 32 megawords. In this case, the test accuracy = 0.156.

6.3 45-unit Realization

The 45-unit realization is shown in Fig. 10. In the circuit, each unit decides if the input image represents the digit i, or the digit j, or another digit. Note that there are $\binom{10}{2} = 45$ units. Each unit produces a **ternary output**: i or j or other digit. A counter counts the number of 1's in the inputs. The counter

	1	2	3	4	5	6	7	8	9
0	**10**	14	14	13	14	15	12	14	14
1		15	15	13	14	13	15	16	13
2			17	15	16	16	17	16	15
3				14	17	14	16	17	16
4					15	14	15	16	**18**
5						17	15	17	16
6							12	16	13
7								15	17
8									17

TABLE 4
Number of compound variables to distinguish a pair of digits in MNIST

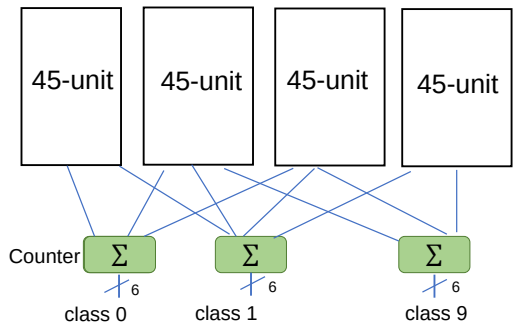

FIGURE 11
45-unit $\times 4$ realization.

with the largest value identifies the most probable input digit. In this case, the test accuracy is 0.901 after SOP minimization [17].

Table 4 shows the numbers of compound variables to distinguish different digits.

When a linear transformation is used, any pair of digits can be distinguished with at most 18 compound variables. This is consistent with the upper bound

$$UP = \lfloor \log_2(k_i k_j + 1) \rfloor = 25$$

given by the Corollary 4.1, where $k_i \simeq 6000$.

In particular, to distinguish digits 0 and 1, we need only 10 compound variables, while to distinguish digits 4 and 9, we need 18 compound variables. This means that to distinguish digits 0 and 1 is the easiest, while to distinguish digit 4 and 9 is the hardest.

6.4 45-unit ×4 Realization

The 45-unit ×4 realization of MNIST is shown in Fig. 11. In this case, the training set is partitioned into four disjoint sets, and each set is implemented by 45-unit. Each counter has $9 \times 4 = 36$ inputs. The total number of units is $45 \times 4 = 180$. In this case, the total amount of memory is smaller than the 45-unit realization. And, the test accuracy is 0.925 after SOP minimization [17].

7 FASHION MNIST

Fig. 12 shows sample images of Zalando's [32] fashion items. Each image is a bit map of $28 \times 28 = 784$ pixels. Originally each pixel was represented by an 8-bit number (i.e., 256-valued grayscale). To reduce the size of data, each pixel was represented by a binary bit, where threshold = 32. Similar to the case of MNIST, the data set consists of 60,000 training images, about 6000 images for each article. There are 10 items. Thus, we have a 784-input 10-output function. Again, this function can be implemented by the circuits in Fig. 9 and Fig. 10, the same architecture as MNIST. When the single-unit realization is used, this function can be represented with $p = 26$ compound variables.

When a 45-input realization is used, any pair of articles can be distinguished with at most 19 compound variables. This is consistent with the upper bound

$$UP = \lfloor \log_2(k_i k_j + 1) \rfloor = 25$$

given by the Corollary 4.1, where $k_i \simeq 6000$.

Table 5 shows the numbers of compound variables to distinguish different articles. In particular, to distinguish (1) Trousers and (7) Sneakers, we need only 5 compound variables, while to distinguish (0) T-shirt/top and (6) Shirt, we need 19 compound variables.

8 COMPARISON WITH NEURAL NETWORKS

In this section, we compare the presented method (**SOP-based method**) with neural networks.

8.1 Realization using Neural Network

Consider the neural network shown in Fig. 13. This circuit was designed by LeCun et. al [8]. It has $n = 784$ inputs, 300 hidden units, and $m = 10$ outputs. The test accuracy is 0.993. In this case, each pixel is represented by 8 bits.

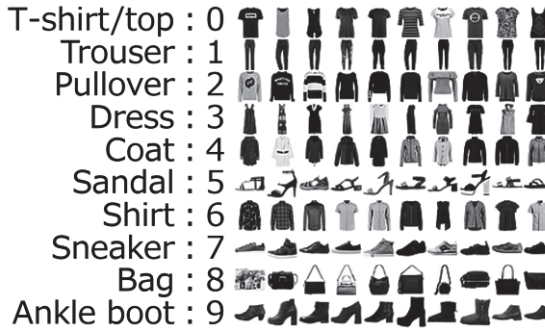

FIGURE 12
Fashion-MNIST data set. Reproduced from an article in ITmedia/@IT with permission [7]. Original data comes from [6], [31].

	1	2	3	4	5	6	7	8	9
0	16	18	18	17	13	**19**	9	16	10
1		14	17	14	9	15	**5**	12	8
2			17	19	10	19	6	17	8
3				18	10	19	7	15	8
4					9	19	6	16	8
5						12	19	16	18
6							8	17	11
7								13	19
8									12

TABLE 5
Number of compound variables to distinguish a pair of articles in Fashion-MNIST.

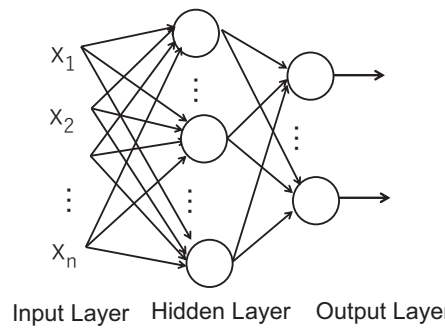

FIGURE 13
Neural network for MNIST

8.2 Comparison of Goals

The goals of two methods are different. The goal of the SOP-based method is to derive a compact set of rules that is consistent with the training set. Thus, the accuracy for the training set is 1.00. On the other hand, the goal of neural networks is to derive a circuit with high accuracy for both the training set and the test set. That is, to derive the circuit with high **generalization ability**. In other words, the goal of the SOP-based method is **logic synthesis** for the training set, while the goals of neural networks is **approximate logic synthesis** [28] [25] for the training set.

The SOP-based method not only memorizes the training set, but also analyzes the data and generalizes it by

- Reduction of variables, and
- Assignment of values to *don't care* elements, which is done during SOP minimization.

Examples are shown in Figs. 5 and 7.

Table 6 compares the SOP-based method with neural networks. The SOP-based method uses an SOP, or a two-level AND-OR circuit, and is implemented by memories. On the other hand, a neural network is a multi-level threshold network. To learn the circuit, the SOP-based method uses reduction of variables, reduction of products, and reduction of literals. On the other hand, neural networks are designed by back-propagation, stochastic gradient descent (SDG), and batch normalization. The SOP-based method is easier and faster to learn. However, the generalization ability is lower than neural networks. The SOP-based method requires a smaller training set than required by neural networks. Also, the SOP-based method is easier to interpret than neural networks. As for the applications, the SOP-based method is suitable for simple problems [18], while neural networks are suitable for complex problems.

9 PLA MEASURE

In the optimization of classification functions, we used two measures: The first measure is the number of variables, and the second measure is the number of the products in an SOP. However, there is a tradeoff between these two measures.

Example 9.1. *Consider the function in Table 7. If we minimize the number of the products in an SOP, it can be reduced to two, but requires four variables, as shown in Fig. 14. On the other hand, if we minimize the number of the variables, it can be reduced to three, but requires three products, as shown in Fig. 15.* ■

	SOP-based method	Neural Network
Circuit Structure	AND-OR two-level	Multi-level threshold network
Method for learning	Reduction of variables Reduction of products Reduction of literals	Backpropagation SGD Batch normalization
Learning	Easy	Difficult
Learning time	Short	Long
Generalization ability	Low	High
Hardware/Power	Low	High
Required training set	Small	Large
Interpretation	Easy	Difficult
Applications	Simple problems	Complex problems

TABLE 6
SOP-based Method vs. Neural Networks

x_1	x_2	x_3	x_4	f
0	1	0	1	1
1	0	0	1	1
1	1	0	0	1
0	0	0	1	0
0	1	1	0	0
1	0	0	0	0

TABLE 7
Partially defined function

As shown in the above example, in a classification function, if we reduce the number of variables first, then the reduction of the number of products in the SOP becomes harder. On the other hand, if we reduce the number of products in the SOP first, then the reduction of the number of variables becomes harder.

To consider both measures at the same time, we introduce the third measure: The area of a programmable logic array (PLA) [11, 20].

Definition 9.1. *The* **PLA measure** *for a two-class function is $n_o \cdot t$, where n_o is the number of variables after optimization, and t is the number of products in the optimized SOP.*

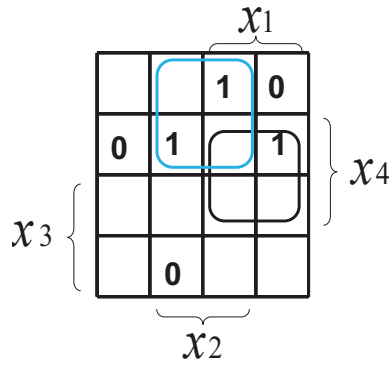

FIGURE 14
SOP with the fewest products: $\mathcal{F}_1$.

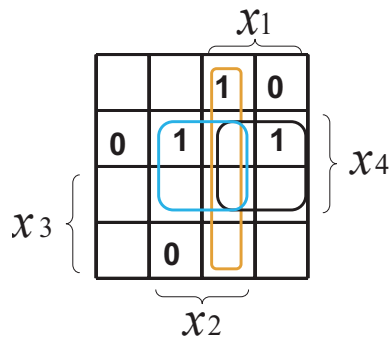

FIGURE 15
SOP with the fewest variables:$\mathcal{F}_2$.

Example 9.2. *The PLA measure for the SOP in Fig. 14 is* $4 \times 2 = 8$, *while that for the SOP in Fig. 15 is* $3 \times 3 = 9$. ■

From experimentalts, optimization of the PLA measure seems to be difficult. For sparse functions, minimization of variables first, and the minimization of the products second, often reduces the PLA measure.

10 CONCLUSION AND COMMENTS

This paper summarized recent results on classification function. It showed that

- Reduction of variables and simplification of SOPs for classification functions improves generalization ability.
- For randomly generated two-class functions with k_0 false minterms and k_1 true minterms, the number of variables can be reduced to $\lceil \log_2(k_0 k_1) \rceil - 2$, in most cases.
- Any two-class function can be represented with at most $\lfloor \log_2(k_0 k_1 + 1) \rfloor$ compound variables.
- For sparse functions, minimization of the variable first, and then minimization of the products often reduces the PLA measure.
- Compared with neural networks, the presented method produces simpler classifiers, but their accuracies are lower.

Applications of the presented approach include ultra-high-speed classifications: packet classification, network intrusion detection, and exotic particle detection in high-energy physics [10, 30]. Other applications include medical diagnosis, where the size of training set is small, and the set is imbalanced [22].

Row-shift decomposition [26] is a new type of a decomposition for a classification function, where the function is implemented by a pair of look-up tables and adder, which is out of scope in this survey.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research of the JSPS. The author thanks Prof. Jon T. Butler for discussion.

REFERENCES

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

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*, Springer, 2006.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. (1987). "Occam's razor," *Information Processing Letters*, 24(6):377–380.

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

[5] P. Domingos. (1999). "The role of Occam's razor in knowledge discovery," *Data Mining and Knowledge Discovery*, 3, 409–425.

[6] GitHub, fashion-mnist zalandoresearch, The MIT License (MIT) Copyright c [2017] Zalando SE, https://tech.zalando.com

[7] https://atmarkit.itmedia.co.jp/ait/articles/2005/28/news016.html

[8] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. (November 1998). " Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 86(11):2278–2324.

[9] http://yann.lecun.com/exdb/mnist/

[10] T. Murovič and A. Trost. (2019). "Massively parallel combinational binary neural networks for edge processing," *Elektrotehniski Vestnik*, 86(1):47–53.

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

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

[13] T. Sasao. *Index Generation Functions*, Springer, Oct. 2019.

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

[15] T. Sasao. (Nov. 8-10, 2020). "On the minimization of partially defined classification functions,' *International Symposium on Multiple-Valued Logic*, (ISMVL-2020), Miyazaki, Japan, 117–123.

[16] T. Sasao, " On the number of variables to represent classification functions using linear decompositions," *Workshop on Synthesis And System Integration of Mixed Information technologies*, (SASIMI-2021), March 29-30, 2021 (Virtual workshop).

[17] T. Sasao, Y. Horikawa, and Y. Iguchi. (Aug. 2021). "Classification functions for handwritten digit recognition," *IEICE Transactions on Information and Systems*, E104-D(8):1076–1082.

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

[19] T. Sasao. "Two-level minimization for partially defined functions," (IWLS-2022), Online, July 18–20, 2022.

[20] T. Sasao. "Data mining using multi-valued logic minimization," *International Symposium on Multiple-Valued Logic*, (ISMVL-2023), Matsue, Japan, May 22-24, 2023.

[21] T. Sasao. "Easily reconstructable logic functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2023), Matsue, Japan, May 22-24, 2023.

[22] T. Sasao, A. Holmgren, and P. Eklund. "A logical method to predict outcomes after coronary artery bypass grafting," *International Symposium on Multiple-Valued Logic*, (ISMVL-2023), Matsue, Japan, May 22-24, 2023.

[23] T. Sasao. *Classification Functions for Machine Learning and Data Mining*,, Springer Nature, Aug. 2023.

[24] T. Sasao. "Iterative linear transformation to reduce compound variables," *Workshop on Synthesis And System Integration of Mixed Information technologies*, (SASIMI-2024), March 11, Taipei, Taiwan.

[25] T. Sasao. (May 29, 2024). "Approximate synthesis of classification functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2024), Brno, Chech Republic, 59–64.

[26] T. Sasao. "Row-shift decompositions for classification functions," *International Workshop on Logic and Synethesis*, (IWLS-2024), June 6, 2024, Zurich, Switzerland.

[27] T. Sasao. (Aug. 2024). "On easily reconstructable logic functions," *IEICE Transactions on Information and Systems*, E107-D(8):913–921.

[28] I. Scarabottolo, et. al. (Dec. 2020). "Approximate logic synthesis: A survey," *Proceedings of the IEEE*, 108(12):2195–2213.

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

[30] Y. Umuroglu, Y. Akhauri, N. J. Fraser and M. Blott. (May 2020). “LogicNets: Co-designed neural networks and circuits for extreme-throughput applications,” *30th International Conference on Field-Programmable Logic and Applications*, 291–297.

[31] H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms,”. arXiv:1708.07747

[32] https://tech.zalando.com