Handwritten Digit Recognition Based on Classification Functions

Tsutomu Sasao Yuto Horikawa Yukihiro Iguchi
Meiji University, Kanagawa, Japan

***Abstract*—As a model of a machine learning, an incompletely specified classification function is used. As a benchmark problem, data for handwritten digits with 28×28 images were used. This data was converted into one with $14 \times 14 = 196$ pixels using a space filter. Also, the value of each pixel was binarized. With this operation, the original data was converted into a 196-variable classification function that takes values from 0 to 9. For the training data, we had $k = 58191$ samples. Using a linear transformation, the 196-variable classification function was converted into a 25-variable function. We applied the testing data consisting of 9569 samples. The reduced classification function produced correct answers for 97.3% of the recognized test data. For unrecognized test data, the circuit for the reduced classification function produced "unrecognized" signals. The recognition circuit for handwritten digits can be implemented by a simple architecture: a cascade of a linear circuit and a memory. To increase the recognition rate, we also present methods using multiple classification functions and voters.**

***Index Terms*—linear decomposition, partially defined function, support minimization, classification, digit recognition, Occam's razor, index generation function, machine learning.**

I. Introduction

Given disjoint sets of elements, the problem to find a simple rule to distinguish these sets, is a major topic of machine learning and data mining. A **partially defined classification function** [12] is the mapping:

$$f : D \to \{1, 2, \ldots, m\},$$

where $D \subset \{0, 1\}^n$ represents the **training set**. When the number of elements in the training set $|D|$ is sufficiently smaller than the total number of input combinations 2^n, the original function f can be represented with compound variables y_j as follows:

$$f(x_1, x_2, \ldots, x_n) = g(y_1, y_2 \ldots y_p), \tag{1.1}$$

where g is a **reduced classification function** of p variables, y_j $(j = 1, 2, \ldots, p)$ are linear functions of the input variable $x_1, x_2, \ldots, x_n$:

$$y_j = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n,$$

where $a_i \in \{0, 1\}$, and $p < n$.

Interestingly, the reduced classification function g produces correct responses not only for the training set, but also for much of unknown **test set**.

That is, the reduced classification function g has a **generalization ability** [2]. The recognition rate of the digits based on reduced classification functions is lower than that of neural networks. However, this method requires no complex learning, and can be implemented by a cascade of a linear circuit and a memory. So, a simple digit recognition is possible.

The reduced classification function correctly recognizes the training set. And for the test set, the function may incorrectly recognize some of input vectors. However, for the real data such as handwritten digits, we demonstrate that the reduced classification function correctly recognizes much part of the test set. The rest of this paper is organized as follows: Section II introduces classification functions; Section III describes compound variables and their reduction method; Section IV explains the benchmark functions; Section V shows the single unit realization; Section VI shows the 10-unit realization; Section VII shows the 45-unit realization; and Section VIII concludes the paper.

TABLE 2.1
REGISTERED VECTOR TABLE

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

II. Definitions

Definition 2.1: Consider the set of k distinct vectors of n bits. These vectors are **registered vectors**. In the framework of machine learning, the set of registered vectors corresponds to the **training set**. To each registered vector, assign an integer between 1 and m, where $2 \leq m \leq k$. The **registered vector table** shows the corresponding **function values** for the registered vectors. A **partially defined classification function** produces the corresponding function values for the input vectors that match the registered vectors. When the input vector does not match a registered vector, the function value is undefined. A partially defined classification function represents a mapping $f : D \to \{1, 2, \ldots, m\}$, where $D \subset B^n$ is the set of registered vectors, and $B = \{0, 1\}$. k is the **weight** of the function. When $m = k$, the function f is an **index generation function** [8], and when $m = 2$, the function f is a **decision function** [10].

Example 2.1: Table 2.1 is the registered vector table of the decision function with weight $k = 6$. ■

III. Compound Variables and Their Reduction

Partially defined functions often can be represented with fewer variables by using **linear decompositions** [11]. In the

978-1-7281-5406-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ISMVL49045.2020.00-18

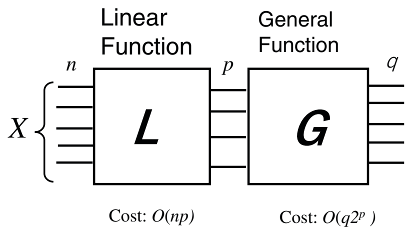

Fig. 3.1. Linear Decomposition

linear decomposition shown in Fig. 3.1, L denotes a linear function, while G denotes a general function (in most cases, non-linear function). We assume the cost of the linear part is $O(np)$, while the cost of the general part is $O(q2^p)$.

Definition 3.1: **Compound variables** have the form $y = c_1x_1 \oplus c_2x_2 \oplus \cdots \oplus c_nx_n$, where $c_i \in \{0,1\}$. The **compound degree** of the variable y is $\sum_{i=1}^{n} c_i$, where $\sum$ denotes an ordinary integer addition, and c_i is an integer. **Primitive variables** are variables with compound degree 1.

Definition 3.2: Given an incompletely specified function f, the linear transformation that minimizes the number of the compound variables is an **optimal transformation**.

When the number of compound variables can be reduced to $q = \lceil \log_2 m \rceil$ by a linear transformation, then the transformation is optimum.

Example 3.1: The function shown in Table 2.1, can be represented as follows:

When primitive variables are used, the function can be represented with only three variables:

$$f = (x_2\bar{x}_4 \vee \bar{x}_1\bar{x}_4) \vee 2(x_2x_4 \vee x_1\bar{x}_2\bar{x}_4)$$

or

$$f = (\bar{x}_2\bar{x}_4\bar{x}_6 \vee x_2\bar{x}_4x_6) \vee 2(x_2x_4 \vee \bar{x}_2\bar{x}_4x_6),$$

where $\vee$ denotes the max operation.

When the compound variables $y_1 = x_4$ and $y_2 = x_2 \oplus x_6$ are used, the function can be represented with only two variables:

$$f = \bar{y}_1\bar{y}_2 \vee 2(y_1 \vee y_2).$$

■

The reduction methods of variables are shown in [12].

IV. Benchmark Functions and Their Evaluation

We use training sets of handwritten digits to find classification functions, and realize them by logic circuits. Since the number of variables and registered vectors is very large, we reduced the sizes of the problems. We considered two problems: The first one consists of 8×8 images, and the second one consists of 14×14 images.

A. 8×8 *Images*

We obtained the function from 32×32 bit maps as follows: First, the maps were partitioned into disjoint blocks of 4×4 images, and the number of non-zero pixels in each block was counted. From this, we had matrices of 8×8, where each element has values in $\{0, 1, 2, \ldots, 16\}$ [14]. To further reduce the size of the data, the matrices were binarized. We set the threshold to 8, and the values of the matrices were converted into $\{0, 1\}$.

TABLE 4.1
Number of 8×8 images.

Data	# of samples
Training Set	3686
Test Set	1675

In this way, we had a 64-variable 10-valued classification function ($n = 64$, $m = 10$). With this operation, some data in the training set and the test set became identical. So, we removed duplicated data. Also, from the test set, we removed data that also appeared in the training set. Table 4.1 shows the sizes of the training set and the test set after this operation.

B. 14×14 *Images*

TABLE 4.2
Number of 14×14 images.

Data	# of samples
Training Set	58191
Test Set	9569

The data in MNIST[15] consists of bit maps of 28×28 images, and the training set consist of 6×10^4 images, and the test set consists of 10^4 images. We partitioned these images into bit maps of 2×2 disjoint blocks, and counted the number of non-zero pixels in each block. With this process, we had a matrix of 14×14, where each element has a value in $\{0, 1, 2, 3, 4\}$. To further compress the data, the matrix was binarized. In this case, we used max pooling. That is, the threshold was set to 1. In this way, we had a 196-variable 10-valued classification function ($n = 196$, $m = 10$). Also in this case, we removed duplicated data. Table 4.2 shows the size of the training set and the test set, after removing duplicated data.

C. *Evaluation of Classifier*

To evaluate the performance of classifier, we use three parameters:

Definition 4.1:

$$\text{Correctness rate} = \frac{\#\text{ of correctly reconized images}}{\text{Total }\#\text{ of recognized images}}$$

$$\text{Recognition rate} = \frac{\text{Total }\#\text{ of recongized images}}{\text{Total }\#\text{ of images}}$$

$$\text{Accuracy} = \frac{\#\text{ of correctly recongized images}}{\text{Total }\#\text{ of images}}$$

Note that the following relation holds among these parameters:

$$\text{Accuracy} = \text{Correctness rate} \times \text{Recognition rate}.$$

In the case of neural nets, accuracy is used to evaluate their performance, rather than the correctness rate or the recognition rate.

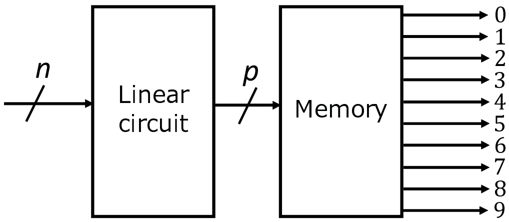

Fig. 5.1. Single-Unit Realization

V. Single-Unit Realization

A **single-unit realization** is implemented by a cascade of a linear circuit and a memory, as shown in Fig. 5.1. When the primitive variables are used, the linear part can be omitted. Thus, it can be implemented by a single memory.

A. 8×8 *Images*

The number of primitive variables for this function was reduced to $p = 21$ by Algorithm 3.1 in [12].

Then, the number of compound variables for this function was reduced to $p = 17$, by the iterative algorithm using Lemma 5.1 in [12].

The reduced classification function g correctly recognized all the training data. Next, we applied the test data shown in Table 4.1 to the reduced classification function g, and checked if g recognized the test data correctly or not. Table 5.1 shows the results. When the digits were unrecognized, the classification function g produced the **unrecognized** output.

TABLE 5.1
Recognition results for single-unit realization (8×8 images).

Result	Primitive variables	Compound variables
	$p = 21$	$p = 17$
Correctly recognized	417	437
Incorrectly recognized	11	32
Unrecognized	1247	1206
Total	1675	1675

Table 5.1 shows that, the recognition rate and accuracy of the reduced classification function g are much lower than that of the neural networks [14], [15], but the function g can be implemented by a cascade of a linear circuit and a memory. Also, no learning is necessary.

B. 14×14 *Images*

The number of primitive variable was reduced to $p = 44$, by Algorithm 3.1 in [12]. Then, the number of compound variables was reduced to $p = 25$, by the iterative algorithm using Lemma 5.1 in [12]. The average and the maximum compound degrees were 3.80 and 12, respectively. Table 5.2 shows the recognition results. When compound variables were used, the correctness rate was 0.973. However, the recognition rate and accuracy are very low.

TABLE 5.2
Recognition result for single-unit realization (14×14 images).

Result	Primitive variables	Compound variables
	$p = 44$	$p = 25$
Correctly recognized	873	1064
Incorrectly recognized	3	29
Unrecognized	8693	8476
Total	9569	9569

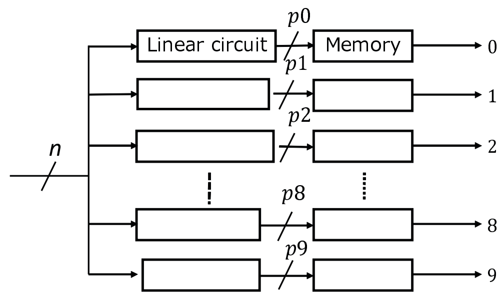

Fig. 6.1. 10-Unit Realization

VI. 10-Unit Realization

In the previous section, the function was implemented by a single unit: a cascade of a linear circuit and a memory. Although the correctness rate was high, the recognition rate and accuracy were low.

Another problem in the single-unit realization is the computation time to reduce the number of compound variables. To reduce the number of variables, the minimization algorithm uses the set of difference vectors, whose size depends on k and m.

The recognition problem of digits is to classify 58191 images into $m = 10$ categories. The circuit shown in Fig. 5.1 solves this problem by a single unit. When the output values are all 0's, the image is **unrecognized**.

The **10-unit realization** uses a separate unit to recognize each of 10 digits, as shown in Fig. 6.1. The top unit decides if the input image is 0 or not; the second unit decides if the input image is 1 or not; $\ldots$, ; and the bottom unit decides if the input image is 9 or not.

Tables 6.1 and 6.2 show the numbers of correctly and incorrectly recognized test images. When s units recognized the image and one of them produced a correct responses, the number of correct answers was counted as $\frac{1}{s}$, and the number of incorrect answers was counted as $1 - \frac{1}{s}$. These tables show that, with multiple units, the recognition rate increases.

TABLE 6.1
Recognition results for the 10-unit realization (8×8 image).

Result	Primitive variables	Compound variables
Correctly recognized	907.5	929
Incorrectly recognized	42.5	63
Unrecognized	725.0	683
Total	1675.0	1675

TABLE 6.2
RECOGNITION RESULTS FOR THE 10-UNIT REALIZATION (14×14 IMAGE).

Result	Primitive variables	Compound variables
Correctly recognized	1779	2147
Incorrectly recognized	17	84
Unrecognized	7773	7338
Total	9569	9569

Table 6.3 shows the recognition result of 8×8 images for each digit. The columnn headed with p_i denotes the number of variables to recognize digit i. Other columns show the number of images with correct responses; the number of images with incorrect responses; and the number of images with unrecognized responses.

Table 6.4 shows the reconition result of 14×14 images.

The recognition rate is higher when compound variables are used. However, the correctness rate is higher when primitive variables are used.

Also, the recognition rates are different for different digits. Especially for 8×8 images, the recognition rate and the correctness rate are high for the digit "0".

VII. 45-UNIT REALIZATION

In the previous section, the i-th unit decides if the input image is i or not. With 10 such units, the recognition rate was improved. In this section, each unit decides if the input image represents i or j or another number. By using $\binom{10}{2} = 45$ such units, we can further improve the recognition rate. Fig. 7.1 shows the **45-unit realization**. Each unit has two

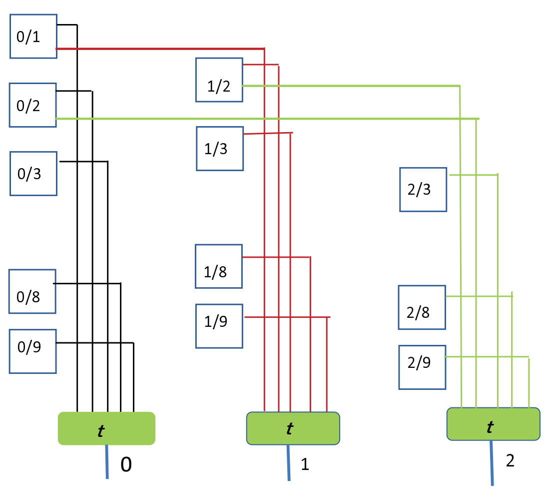

Fig. 7.1. 45-unit realization.

outputs: The output $(1, 0)$ denotes that the input image is i; the output $(0, 1)$ denotes that the input image is j; and the output $(0, 0)$ denotes that the input image is another number or unknown. Since there are 45 units, the total number of outputs is 90. In addition, we use 10 threshold elements (or voters). The i-th threshold element has 9 inputs with label i, and produces one if and only if the number of active inputs is

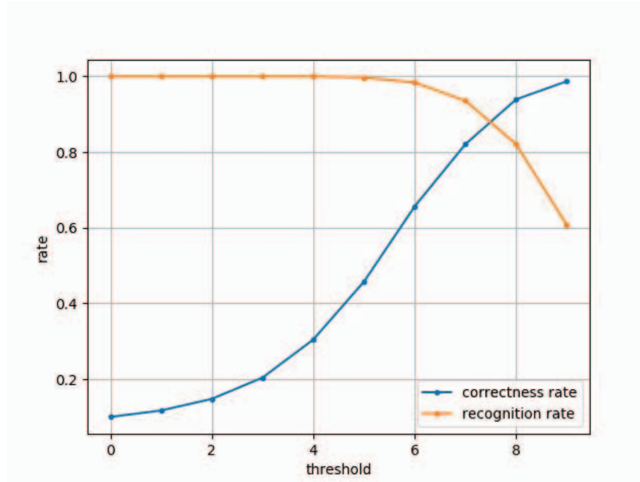

Fig. 7.2. Correctness rate and Recognition rate vs. threshold t, (45 units, primitive variables: 8×8 image.)

equal to or greater than t. By appropriately selecting the value of the threshold t, we can improve the recognition rate and/or correctness rate.

Fig. 7.2 shows correctness rate and recognition rate for different values of threshold t. This data is for 8×8 images and only primitive variables are used to recognize the data. With the increase of threshold t, the correctness rate increases, while the recognition rate decreases. When $t = 7$ and $t = 8$, both the recognition rate and correctness rate are more than 0.8.

In many cases, the accuracy takes its maximum when the threshold is set so that the correctness rate is equal to the recognition rate. For example, Fig. 7.2 shows that the accuracy takes its maximum when $t = 7$ or $t = 8$. The accuracy of 45 units are summarized in Table 7.2.

Table 7.1 and Table 7.3 show recognition results for four different implementations. In all cases, by selecting appropriate thresholds, we could improve the accuracy.

Table 7.4 and Table 7.5 show average number of variables and its standard deviations. Note that the number of variables is smaller than those of the 10-unit realizations. Thus, the amount of memory for each unit is also smaller.

The total amount of memory to implement the 45-unit realization is

$$\sum_{i=0}^{8} \sum_{j=i+2}^{9} \lceil \log_2(2+1) \rceil 2^{p_{ij}},$$

where p_{ij} denotes the number of the variables for the unit (i, j).

Table 7.6 compares the memory sizes of circuits for 8×8 images, while Table 7.7 compares the memory sizes of circuits for 16×16 images. The memory sizes for 9-input voting functions are not included. Note that an 9-input voting function can be implemented by two 6-input LUTs.

These tables show that the 45-unit realization requires less hardware, with higher accuracy.

TABLE 6.3
RECOGNITION RESULTS FOR THE 10-UNIT REALIZATION (8×8 IMAGE).

	Primitive variables				Compound variables			
Digit	p_i	Correctly recognized	Incorrectly recognized	Unrecognized	p_i	Correctly recognized	Incorrectly recognized	Unrecognized
0	11	130	2.0	13	11	129.0	4.0	12
1	17	55.5	5.5	75	14	59.5	7.5	69
2	13	114	4.0	56	12	115.5	5.5	53
3	16	83	4.0	91	14	84.0	5.0	89
4	16	98	5.0	71	13	101.0	6.0	67
5	15	109	4.0	62	14	110.0	7.0	58
6	14	120	3.0	42	12	127.5	3.5	34
7	13	118	1.0	59	12	116.0	3.0	59
8	18	24	7.0	141	15	27.0	12.0	133
9	18	56	7.0	115	14	59.5	9.5	109
Total		907.5	42.5	725		929.0	63.0	683

TABLE 6.4
RECOGNITION RESULTS FOR THE 10-UNIT REALIZATION (14×14 IMAGE).

	Primitive Variables				Compound Variables			
Digit	p_i	Correctly recognized	Incorrectly recognized	Unrecognized	p_i	Correctly recognized	Incorrectly recognized	Unrecognized
0	29	317.0	3.0	660	21	415.5	5.5	559
1	37	389.5	0.5	320	21	443.0	2.0	265
2	30	127.0	1.0	904	22	136.0	13.0	883
3	33	42.0	1.0	967	22	65.0	8.0	937
4	34	135.5	2.5	844	22	159.0	9.0	814
5	31	66.0	1.0	825	22	87.0	13.0	792
6	30	382.0	3.0	572	21	398.5	5.5	553
7	34	206.0	4.0	817	22	284.5	10.5	732
8	37	17.0	0.0	957	22	41.0	6.0	927
9	36	97.0	1.0	907	23	117.5	11.5	876
Total		1779.0	17.0	7773		2147.0	84.0	7338

TABLE 7.1
RECOGNITION RESULT FOR 45-UNIT REALIZATION (8×8 IMAGE).

Result	Primitive variables $t = 8$	Compound variables $t = 8$
Correctly recognized	1295.50	1324.0
Incorrectly recognitized	83.50	106.0
Unrecognized	300.00	245.0
Total	1675.00	1675.0

TABLE 7.2
ACCURACY OF 45-UNIT REALIZATIONS.

Image	Primitive variables	Compound variables
8×8	0.771 $t = 8$	0.790 $t = 8$
14×14	0.700 $t = 3$	0.720 $t = 4$

TABLE 7.3
RECOGNITION RESULT FOR 45-UNIT REALIZATION (14×14 IMAGES).

Result	Primitive variables $t = 3$	Compound variables $t = 4$
Correctly recognized	6700.95	6889.12
Incorrectly recognized	1551.05	1247.88
Unrecognized	1317.00	1432.00
Total	9569.0	9569.00

TABLE 7.4
NUMBER OF VARIABLES FOR 45-UNIT REALIZATION (8×8 IMAGES).

	Primitive variables	Compound variables
Average	7.51	7.00
Standard Deviation	2.15	1.74

TABLE 7.5
NUMBER OF VARIABLES FOR 45-UNIT REALIZATION (14×14 IMAGES).

	Primitive variables	Compound variables
Average	20.98	16.33
Standard Deviation	3.30	1.51

TABLE 7.6
MEMORY SIZES OF CIRCUITS FOR 45-UNIT REALIZATION (8×8 IMAGES, MEGA BITS).

8×8	Single Unit	10 Units	45 Units
Pririmive Variables	8.4	0.9	0.07
Compound Variables	0.5	0.1	0.02

TABLE 7.7
MEMORY SIZES OF CIRCUITS FOR 45-UNIT REALIZATION (14×14 IMAGES, MEGA BITS).

14×14	Single-Unit	10-Unit	45-Unit
Pririmive Variables	$70,368,744.2$	$391,378.9$	$3,788.8$
Compound Variables	134.2	39.8	11.8

VIII. Concluding Remarks

In this paper, we showed that classification functions are useful for recognition of handwritten digits.

The reduced classification function g recognized correctly not only for all the training data, but also for much of the test data. The reason why the reduced classification functions g have the generalization capability, can be explained as follows: During the minimization of variables, necessary variables to recognize the digits are selected. This process corresponds to the **feature extraction** of digits in the training data.

In the framework of probably approximate correct learning [16], Occam's razor is known. Occam's razor recommends using as simple rules as possible [1]. With this strategy, we can expect that *the fewer the variables to distinguish digits, the higher the accuracy for the test data*. Also, when the number of the variables are the same, we can expect that *the smaller the compound degree, the higher the accuracy*.

First, the number of primitive variables was reduced, and the second, the number of compound variables was reduced. When the values of k and m are large, the computation time and necessary amount of memory increased rapidly.

To reduce the values of k and m, a separate unit was used to recognize each digit. In the 10-unit realization, the decision function for each unit became simpler, and the accuracy increased considerably. Also, we found that the recognition rates are different for different digits. We also showed the 45-unit realization. Although it is more complicated, the accuracy was improved by selecting an appropriate threshold.

Accuracy of the circuits derived from classification function is lower than that of neural networks. However, the amount of necessary hardware is much smaller. Also, no learning is necessary. In the case of 8×8 images, just a single memory with 17 inputs, and a small amount of hardware for the linear circuit are sufficient.

One of the reviewer pointed out that when s units recognize the image, the system cannot find the correct answer. So, in such a case, it should be considered as *unrecognized*. This problem can be solved by using extra hardware to choose the answer randomly. In such a case, the correctness rate will be $\frac{1}{s}$. To implement it by a combinational circuit, a priority encoder would be a simpler and realistic.

Most neural nets for MNIST use *soft max* functions in the output layer. Note that each of outputs denotes the probability (possibility) p_i, where $\sum_{i=0}^{9} p_i = 1.0$. For example, if the probabilities for the digits "0" and "1" are both 0.5, and probabilities for other digits are 0.0, then the correctness rate is computed as 0.5. Also in the case of neural nets, the circuit to find the output with the largest probability is not included.

As for the generalization ability for logic circuits, [5] and [2] also consider it. However, [5] used random logic circuits, while [2] used multi-level LUT networks. Thus, they are more complicated.

Acknowledgments

This research is partly supported by the grant of the Japan Society for the Promotion of Science (JSPS), Grant in Aid for Scientific Research. Dr. Alan Mishchenko of University California, Berkeley, and Dr. Satrajit Chatterjee of Google AI gave us useful comments. Reviewers comments were also useful. Prof. Jon T. Butler improved English presentation.

References

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

[2] S. Chatterjee,"Learning and memorization," *International Conference on Machine Learning (ICML 2018)*, Stockholm, Sweden, July 10-15, 2018, pp. 754-762.

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

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

[5] A. L. Oliveira and A. Sangiovanni-Vincentelli, "Learning complex boolean functions:Algorithms and applications," *Advances in Neural Information Processing Systems*, No. 6, pp. 911-918. Morgan-Kaufmann, 1994.

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

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

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

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

[10] T. Sasao, "On a minimization of variables to represent sparse multi-valued input decision functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2019), Fredericton, Canada, May 21-23, 2019, pp. 182-187.

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

[12] T. Sasao, "On the minimization of variables to represent partially defined classification functions," *International Symposium on Multiple-Valued Logic*, (ISMVL-2020), May 20-22, Miyazaki, Japan.

[13] D. A. Simovici, M. Zimand, and D. Pletea,"Several remarks on index generation functions,"*International Symposium on Multiple-Valued Logic* (ISMVL-2012), Victoria, Canada, May 2012, pp. 179-184.

[14] https://archive.ics.uci.edu/ml/datasets /optical+recognition+of+handwritten+digits

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

[16] L. G. Valiant,"A theory of learnable," *Communications of the ACM*, Vol. 27, No. 11,pp. 1134-1142, 1984,