A Self-Organizing Binary System*

RICHARD L. MATTSON†

Introduction

ANY STIMULUS to a system such as described in this paper can be coded into a binary representation. A character on a piece of paper can be represented in binary form by dividing the paper into many small squares and assigning a 1 to a square if the character covers that square, and a 0 to the square if the character does not cover it.

A voltage waveform can be coded into a binary representation by quantizing the waveform in both time and amplitude, and assigning a binary code to each amplitude. In general, any data that are measurable can be coded into a binary representation with no loss of information. One form of an interesting logical system would be one which transformed this block of binary data into a single binary decision. For example, such a system would take the binary representation of a character and transform it into a binary decision (either it is an "A" or it is not an "A"). Combinations of such devices could then be used to make decisions of a higher order than binary.

The conventional method of designing such a system is to list in a table all possible combinations of the binary inputs, and to assign the most probable output value (decision) to each input combination. This table of combinations with output values is then reduced to a Boolean function which represents the method by which the decision is to be determined. This procedure will always yield the best possible technique of making decisions.

However, this synthesis procedure has two major disadvantages. First, the Boolean function is not easily obtained when the number of binary variables at the input is large. Second, even if the Boolean function were determined, any change in the decision process would require a repetition of the difficult synthesis procedure in order to arrive at a new Boolean function.

It is the purpose of this paper to define a model for a self-organizing logical system which determines, by an iterative trial-and-error procedure, the proper Boolean function for a process. This self-organizing logical system is composed of two separate systems, one a network of adjustable logical devices, the other a system for determining what logical function should be assigned to each device in the network. Such a system is depicted schematically in Fig. 1.

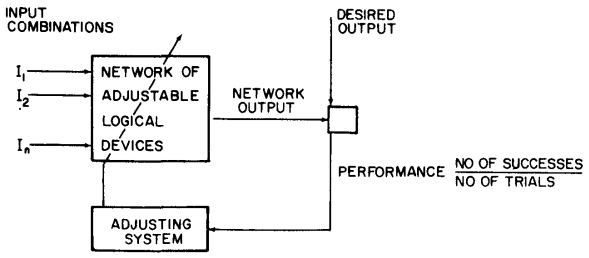

Fig. 1—Self-organizing logical system.

The adjusting system makes a sequence of adjustments in the network, each adjustment being determined by the performance of the network for the previous adjustments, so that with the final adjustment the network realizes the desired Boolean function.

An Adjustable Logical Device

A logical device that can realize any one of many different logical functions by adjusting its internal parameter values is shown schematically in Fig. 2.

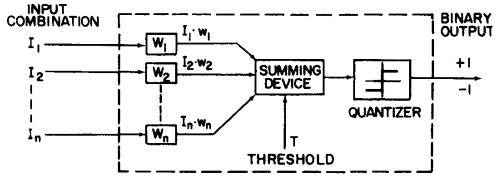

Fig. 2—The adjustable logical device.

The inputs to this device are binary, + 1 and − 1. The weights, w_i, are continuous real variables that can assume either positive or negative values. The threshold, T, is also a continuous real variable and can be either positive or negative. The output of this device is + 1 if

$$T + \sum_{i=}^{n} I_i w_i > 0 \qquad (1)$$

and − 1 if

$$T + \sum_{i=}^{n} I_i w_i < 0 \qquad (2)$$

The dividing condition between the output being + 1 or − 1 is given by

$$T + \sum_{i=}^{n} I_i w_i = 0 \qquad (3)$$

* Most of the work described in this report, including computer simulation studies, was done at the Digital Systems and Components Laboratory and the Computation Center, Massachusetts Institute of Technology, under Navy BuShips Contract NObsr 72716. The work was completed at Lockheed Missiles and Space Division, Logical Design, as a part of the Lockheed General Research Program.

† Missile Systems Division, Lockheed Aircraft Corporation, Sunnyvale, Calif.

A geometric representation of the logical properties of this device can be made by using the equation of the dividing condition (Equation 3), and a binary n-cube representation of the input space of the device. For example, with three inputs to the model, the dividing condition is

$$T + I_1w_1 + I_2w_2 + I_3w_3 = 0$$

By plotting this plane in an I_1, I_2, I_3 space, and indicating various input combinations by points in the space, the logical operation of the device can be interpreted geometrically. That is, all points on one side of the plane are mapped into a + 1 by the device, and all points on the other side are mapped into a − 1. A plot of the input space is shown in Fig. 3. In an n-dimensional input space the dividing condition specifies a hyperplane which separates the n-dimensional input space into two parts. It can easily be shown that the w_i values control the slope of the hyperplane in the space, and the threshold, T, controls the position of the plane in the space.

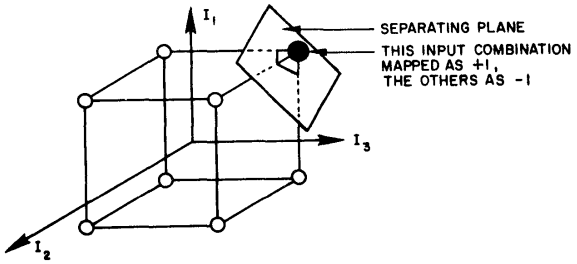

Fig. 3—Three-variable input space.

By changing the w_i values and T, different mapping functions can be obtained. For example, with this device one can obtain 14 different mapping functions for two-input variables, 104 for three-input variables, and approximately 1900 for four-input variables. These different mapping functions are termed "linearly separated" truth functions[2]. The idea of passing a hyperplane through an n-dimensional input space is important because it allows one to visualize types of functions that can be generated by the logical device. It is also useful for determining the types of processes that the model would be best suited to classify. For example, since the hyperplane divides the input space into two regions, each of which is a region of points unit-distant apart, the logical device would be useful in classifying processes in which one group of unit-distant points is separated from another group of unit-distant points. An example of such a process is a single-error correcting code where a legal n-bit word, and all combinations unit-distant from this word, are to be mapped as the legal word. Another possible use would be for character recognition where a legal character is to be separated from all other characters. For this application each character and its variations must be representable as a group of closely connected points in the input space, and each of these groups must be separable from all the other groups. When this condition is met, a single logical device would be an effective character classifier.

Another useful capability of the device is that of assigning output values to input combinations that have never been given to the device. For example, if a small number of input combinations have been given to the logical device and a hyperplane is placed in the input space so as to map these combinations correctly, then every point in the input space is automatically assigned an output mapping by this plane. This allows a mapping criterion to be established without being exposed to all possible input combinations. In the case of character recognition, where the number of possible variations of a legal character is so large that it becomes impossible to list all of them, a hyperplane could be situated in the input space on the basis of a partial listing of variations, and the mapping of the other variations would automatically be determined.

By interconnecting devices of this type, each one realizing a different linearly separated function, and feeding their outputs into an AND or OR device, any logical function can be obtained. This is easily shown since a single device of this type can realize the AND or OR function of n-variables and any Boolean function can be written as a sum of products or product of sums.

A network of adjustable logical devices can be constructed by using as a basic element the device shown in Fig. 2. By adjusting various weights and thresholds in the network any logical function can be realized. The remaining portion of this paper describes methods of determining what sequence of network adjustments should be made to obtain convergence to a desired Boolean function.

Analysis of the Performance of a Single Logical Device

Consider the input space of an n-variable logical device. There will be 2^n distinct points in this input space, each one corresponding to a particular input combination of the n-variables. Let these combinations be denoted as $C_1, C_2, \ldots, C_k \ldots, C_{2n}$. Each of these input combinations has a probability, $P(C_k)$, of occuring at the input of the logical device. For each input combination the desired output has a probability of being + 1, $P(D_k = +1 \mid C_k)$, and a probability of being − 1, $P(D_k = -1 \mid C_k)$. If the input combination C_k is mapped as a + 1 by the device, then the probability that the output of the device will agree with the desired output is

$$P_k(A) = P(D_k = +1 \mid C_k) \qquad (4)$$

If C_k is mapped as a -1 by the device, the probability of agreement between the desired output and the output of the logical device is

$$P_k(A) = P(D_k = -1 \mid C_k) \tag{5}$$

Since C_k must be mapped either as a $+1$ or a -1, then one of the two above equations is valid for the probability of agreement for a fixed logical device, and the other is not. Define a variable x_k so that $x_k = 1$ when C_k is mapped as a $+1$ and is *zero* otherwise, and a variable $\bar{x}_k$ which is $+1$ when C_k is mapped as a -1 and is *zero* otherwise. The probability of agreement for input C_k can then be written as

$$P_k(A) = x_k P(D_k = +1 \mid C_k) + \bar{x}_k P(D_k = -1 \mid C_k) \tag{6}$$

The total probability of agreement between the output of the logical device and the desired output can be obtained by summing the individual performances of all the input combinations.

$$P(A) = \sum_{k=1}^{2^n} [x_k P(D_k = +1 \mid C_k) + \bar{x}_k P(D_k = -1 \mid C_k)] \tag{7}$$

The performance of the logical device is measured by the number of agreements divided by the number of trials. The expected value of this measurement is given by $P(A)$, and thus $P(A)$ is the expected performance of the logical device.

Earlier it was pointed out that the weights in the logical device control the slope of the hyperplane in the input space of the model, while the threshold controls the location of the plane in that space. Thus, with the threshold set at an extremely negative value the hyperplane will lie outside the binary n-cube in that space, and all points in the input space will be mapped as -1. As T is increased in value the hyperplane will pass through the binary n-cube, changing the mapping of the points in space to $+1$ until T is extremely positive and all points are mapped as $a + 1$. With T extremely negative all points in the input space are mapped as -1 and Equation 7 becomes

$$P(A) = \sum_{k=1}^{2^n} P(D_k = -1 \mid C_k) \tag{8}$$

If the index k is arranged so that the sequence $k = 1, 2, 3, \ldots, 2^n$ corresponds to the first, second, $\ldots, 2^n$th point in the input space to have its mapping changed to $+1$ as T is increased, then the performance will have the sequence of values.

$$P_0(A) = \sum_{k=1}^{2^n} P(D_k = -1 \mid C_k)$$

for T extremely negative

$$P_1(A) = P(D_1 = +1 \mid C_1) + \sum_{k=2}^{2^n} P(D_k = -1 \mid C_k)$$

increasing T ↓

$$P_2(A) = \sum_{k=1}^{2} P(D_k = +1 \mid C_k) + \sum_{k=3}^{2^n} P(D_k = -1 \mid C_k)$$

$$\vdots$$

$$P_{2^n}(A) = \sum_{k=1}^{2^n} P(D_k = +1 \mid C_k) \text{ for } T \text{ extremely positive}$$

as T is increased from negative to positive. In addition, if $P_0(A) < P_1(A)$, then $P(D_1 = -1 \mid C_1) < P(D_1 = +1 \mid C_1)$ and C_1 should be mapped as $a + 1$. By comparing $P_1(A)$ with $P_2(A)$ etc., it can be shown that if the performance continuously increases as T is increased, points in the input space are continuing to be mapped correctly as $+1$. Whenever the performance decreases for increased T, that point should not be mapped as $+1$. From this result it is possible to prove that if $P(A)$ increases monotonically to a maximum (peak) value and decreases monotonically thereafter, then the mapping function corresponding to T set at the peak is the best possible mapping function for that particular input space. This is easily shown since, for T set at the peak, the performance is

$$P(A) = \sum_{K=1}^{i} P(D_k = +1 \mid C_k) + \sum_{K=i+1}^{2^n} P(D_k = -1 \mid C_k)$$

and for each $1 \leq k \leq i$

$$P(D_k = +1 \mid C_k) > P(D_k = -1 \mid C_k)$$

and for each $i + 1 \leq k \leq 2_n$

$$P(D_k = -1 \mid C_k) > P(D_k = +1 \mid C_k)$$

By using similar arguments it can be shown that if there exist m groups of $+1$ points in the input space that are separated from each other by a group of -1 points, there will be at least m peaks on the performance curve. A sample performance curve vs. threshold for $m = 1$ is shown in Fig. 4.

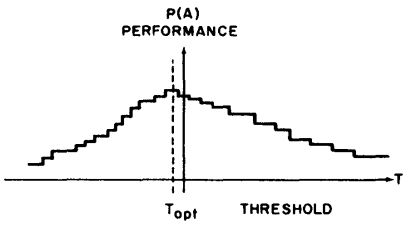

Fig. 4—Performance curves with one peak.

Equation 7 gave the performance of the logical device in terms of the performance for each input combination. The index k was associated with the sequence of points that change their mapping function as T is increased. When a given weight, w_i, is changed, the hyperplane will rotate and another sequence of points will change their mapping func-

tion. Thus, the same results that applied for the threshold can also be applied for each weight of the device, and a curve of the performance as a function of w_i can be plotted. Again it can be shown that as the performance increases a group of points in the input space is being correctly mapped, and as soon as the performance decreases a point is being incorrectly mapped by the change in parameter of the device.

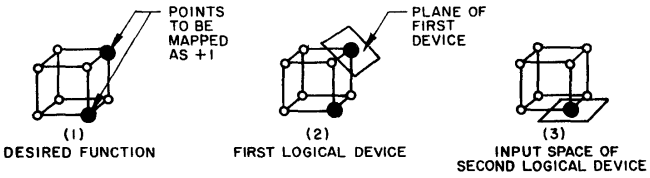

Fig. 5—Synthesis with two logical devices.

The addition of other logical devices can be accomplished by letting each one adapt to a group of separated points in the input space. For example, consider the input space shown below. Here, since there are separated groups in the input space, a single logical device cannot realize the desired function. However, a single logical device can realize the function shown in Fig. 5-(2). If the performance curve of this logical device is plotted against its threshold it will have two peaks. One peak will correspond to the function shown in Fig. 5-(2) and this logical device should be adjusted to realize that function. Every time the first logical device has an output of $+1$ the desired output of the second logical device should be -1. This makes the input space of the second logical device be as shown in Fig. 5-(3). Clearly this is a linearly separated function and can be realized by the second logical device. By feeding the outputs of these two devices into an OR function, the desired function is obtained. It can be shown that the most difficult logical function to mechanize in this manner is the alternate symmetric or even-odd functions. In this case it requires 2^{n-1} logical devices all feeding into an OR gate to realize the function.

An Adjustment Procedure

Having related the performance of the device to the change in parameter values, an adjustment procedure can be devised for the adjusting system. Briefly, the procedure is to start with a single logical device and make a guess at initial values for weights in the device. It has been demonstrated experimentally and justified mathematically that good results are obtained if each weight is initially set to the value of the cross-correlation between the particular input and the desired output.[1] This quantity is computed from the equation

$$w_k = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I_i D_j P(I_i, D_j) = P(I_k = D) - P(I_k = D)$$

For certain processes this initial setting is not the best one to make; however, it is good for the types of processes considered in the demonstrations. Then vary the threshold from extremely negative to less negative values and measure the performance for each value of T. If the performance increases, increase T and measure the performance again. If the performance decreases adjust the threshold back to the peak. Then increase the weights one at a time until a peak is obtained for each weight. The threshold and weights should be re-adjusted until any change in any parameter causes a decrease in performance. At this point the hyperplane has a group of points on one side that are correctly mapped, and all points closest to the other side of the plane require the opposite mapping. Then T should be increased to an extremely positive value and a measure of the performance recorded for each value of T. If the performance monotonically decreases for increasing T then the single logical device with T set at the peak realizes the desired truth function for the process. If the performance curve has another peak as T is increased, then a single logical device cannot realize the desired function, and other logical devices must be added.

Because the adjusting system obtains only a measure of the $P(A)$ function, the measured performance may deviate from the theoretical performance. When this happens the performance curve will not look like the one shown in Fig. 4, but rather will have many small peaks and valleys as shown in Fig. 6. On this measured performance curve a slight decrease in the measured performance may be due to sample-size effects of measuring. Therefore confidence limits must be established by the adjusting system to determine if the performance has actually decreased, or if the decrease is due to measurement errors. Thus, statements about the performance of the logical system must be modified to include confidence limits. As an example, with 95 percent confidence the performance curve has only one peak and therefore with 95 percent confidence the logical system is realizing the best possible function for this process.

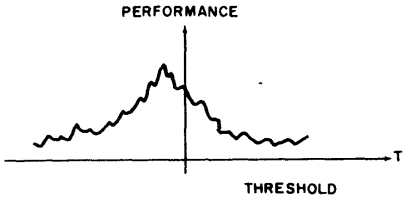

Fig. 6—Measured performance curve.

The process of determining a peak on the performance curve can be speeded up by assuming a slowly changing curve and by selectively sampling the curve. Consider the curve shown in Fig. 7. This

curve is first sampled at $n + 1$ equally-spaced values of T. Because the curve increases monotonically to a peak and then decreases, one peak must be between the first and third sampling. The curve is then sampled n times in this region. The peak must now be between the second and sixth samples so the curve is sampled n times in this region, and this process is repeated until the peak has been determined. This process can be shown to converge geometrically to the peak.

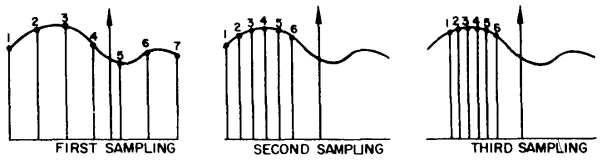

Fig. 7—Successive sampling of the performance curve.

Pattern Recognition Demonstration [3,4]

The devices and the adjusting system were simulated on the IBM 704 computer and four demonstrations of the self-organizing logical system were made. Three of these were character-recognition problems and one was waveform recognition. In the character-recognition problems the network of logical devices was to be adjusted until it realized an optimum truth function for distinguishing one group of characters from another group in the presence of noise. The initial settings of the weights was the cross-correlation and in each demonstration this was a setting which gave optimum performance. In each case the noise-free characters were arranged in a fixed $m \times n$ array of squares just large enough to contain the characters. Noise was introduced into the characters by selecting each of the squares in the array, one at a time; the representation of each square was changed with a probability, p, and was left unchanged with a probability, $1 - p$.

Demonstration 1: Recognize 5 and S; 25 Binary Variables

In this problem the system was required to distinguish a 5 and a shifted 5 from an S and a shifted S in the presence of noise. The noise-free and noisy characters are shown in Fig. 8.

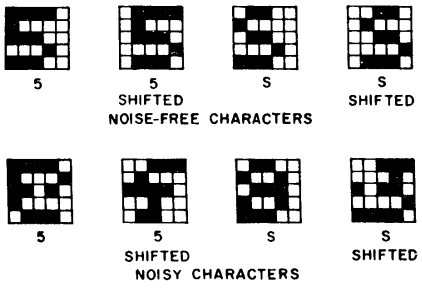

Fig. 8—Characters in Demonstration 1.

With 1.5 minutes of computer time, 2192 independent samples of noisy characters, and a single logical device the 5's were distinguished from the S's with 98 percent recognition. Because of the noise this was the best possible theoretical performance for this process. The performance curves were sampled three times to determine the best setting of the threshold. These curves are shown in Fig. 9. Since the adjusting system used a confidence limit of 99.5 percent, it is 99.5 percent certain that the self-organizing system was realizing the best Boolean function for this process.

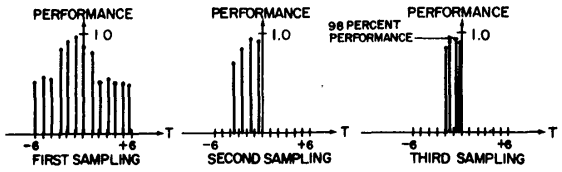

Fig. 9—Sampling of performance curve, Demonstration 1.

Demonstration 2: Recognize 5, S and 8; 49 Variables

In this problem the system was required to distinguish a 5 from an S and an 8 arranged in a 7×7 array. Noise was introduced in the same fashion as in Demonstration 1. The noise-free and noisy characters are shown in Fig. 10.

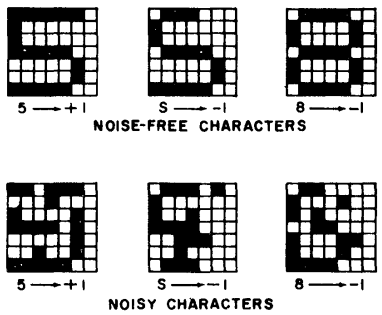

Fig. 10—Characters in Demonstration 2.

It required 1.6 minutes of computer time, 1337 samples of noisy characters, and one 49-input logical device to realize an optimum truth function for this recognition problem. The sampled performance curves for this process are shown in Fig. 11, and again there was 99.5 percent certainty of realizing the best possible Boolean function.

Demonstration 3: Recognize OF, AT, TO, and IN; 50 Variables

In this problem the system was required to distinguish the word *OF* from the words *AT*, *TO*, and *IN* displayed in a 10×5 array. Noise was introduced in the same manner as before. The noise-free and noisy characters are shown in Fig. 12.

It required 1.1 minutes, 718 samples of noisy characters, and a single 50-input logical device to realize

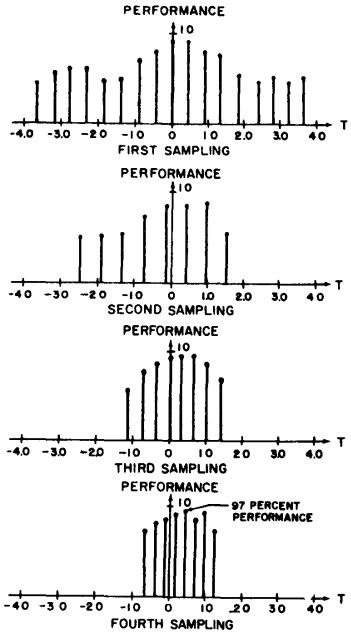

Fig. 11—Sampling of performance curve, Demonstration 2.

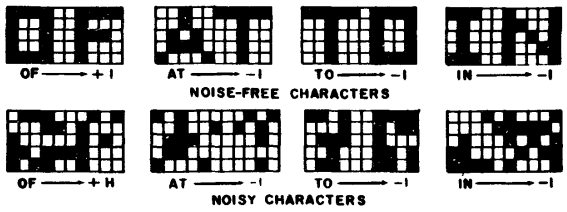

Fig. 12—Characters in Demonstration 3.

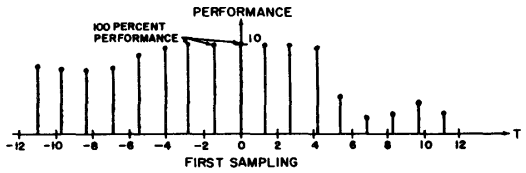

Fig. 13—Sampling of performance curve, Demonstration 3.

the optimum function (100 percent recognition) for this problem. The single sampled performance curve is shown in Fig. 13.

Demonstration 4: Recognition of Waveforms

The fourth demonstration problem for the self-organizing logical system was to distinguish one waveform from a similar waveform in the presence of noise. The two waveforms were quantized into 36 discrete instants of time and the amplitude at each instant was coded into a 6-bit binary code. Thus, the binary representation of a waveform required the combination of 216 binary bits. Noise was introduced so that the greatest amount of noise appeared in the least significant bits of the amplitude code, and the least amount of noise appeared in the most significant bits of the amplitude code. Examples of noise-free and noisy waveforms are shown in Fig. 14.

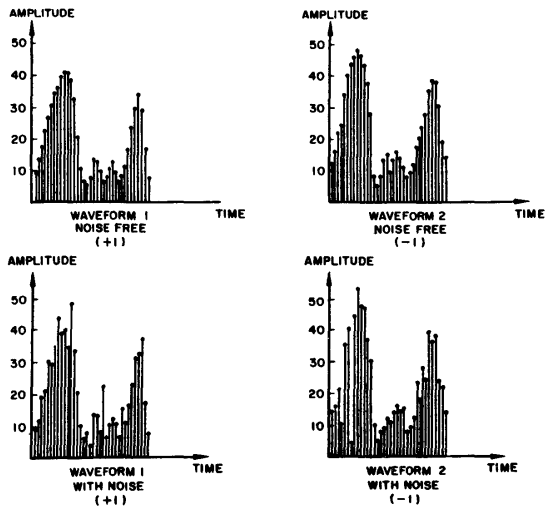

Fig. 14—Waveforms, Demonstration 4.

The self-organizing logical system required 0.9 minutes, 987 samples of noisy waveforms, and a single 216-input logical device to distinguish the waveforms with 100 percent recognition. A sampling of the performance curve is shown in Fig. 15.

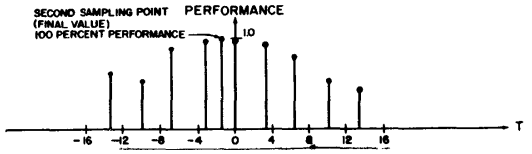

Fig. 15—Sampling of performance curve, Demonstration 4.

Conclusions

From the results obtained in the simulation studies, it appears that this technique of self-organization allows solution of many practical pattern-recognition problems. These problems could take a variety of forms, from visual recognition to speech recognition, and more complicated data reduction problems such as binary predicting and filtering in the presence of noise. The time and amount of data required for the system to "adapt" to a given arbitrary task seem to be within practical limits. It is not known what the equipment requirements would be for a general self-organizing system, and a variety of more difficult pattern-recognition problems are needed to determine the versatility of a "self-organizing" logical system requiring many logical devices.

References

[1] R. L. Mattson, *The Design and Analysis of an Adaptive System for Statitiscal Classification*, S. M. Thesis, M.I.T., Course VI, June 1959

[2] Applied Mathematics and Statistical Laboratory, *Unate Truth Functions*, by Robert McNaughton, Technical Report No. 4, Stanford University, Oct. 21, 1957.

[3] Computer Components and Systems Laboratory, *An Adaptive Classifier*, by R. L. Mattson, Quarterly Progress Report No. 6, Cambridge, M.I.T., March 1959.

[4] Computer Components and Systems Laboratory, *An Adaptive Classifier*, by R. L. Mattson, Quarterly Progress Report No. 7, Cambridge, M.I.T., June 1959.