A GENERALIZED LEARNING NETWORK
USING ADAPTIVE THRESHOLD ELEMENTS

by

Haruhisa Ishida

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major Subject: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Head of Major Department

Signature was redacted for privacy.

Dean of Graduate College

Iowa State University
Of Science and Technology
Ames, Iowa

1964

TABLE OF CONTENTS

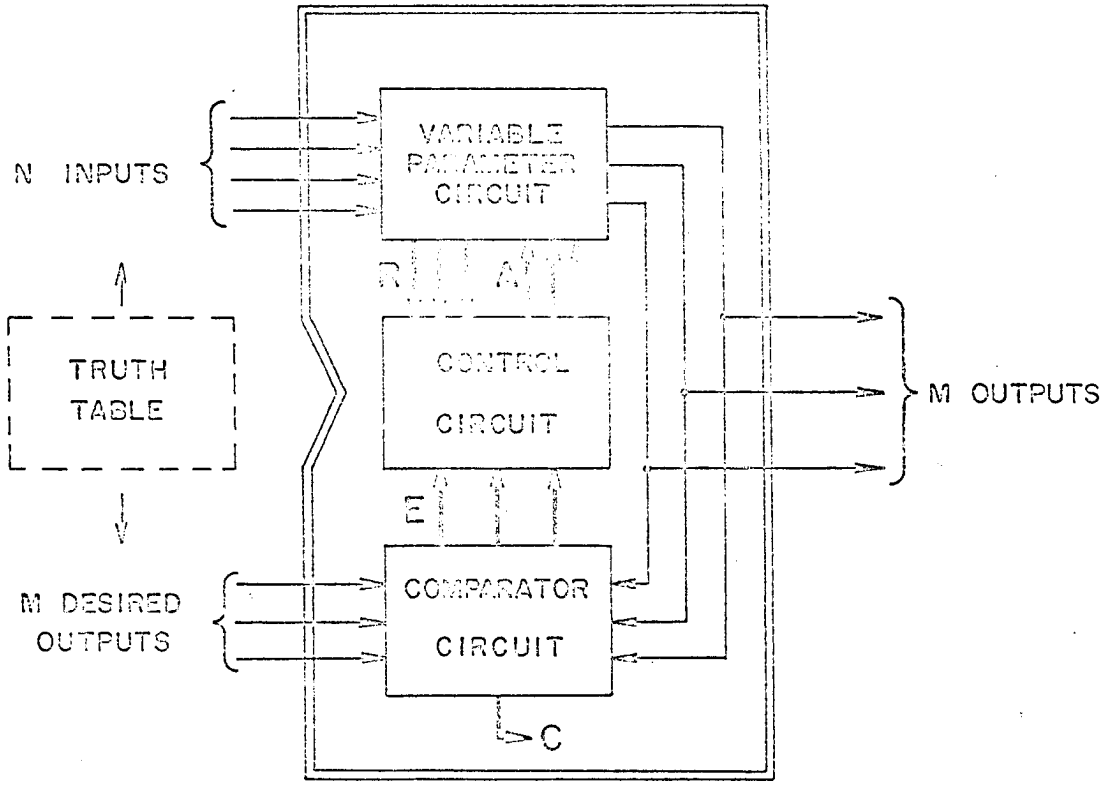
I. INTRODUCTION

In the course of years since the advent of the first electronic computer, ENIAC, in 1946, the computer has acquired greater capabilities due to its higher operation speed, larger memory capacity and increased versatility of its organization. Yet as originally intended, the computer is still primarily suited for handling computational problems. It is not always as effective for non-arithmetical jobs such as language translation, pattern recognition, etc., because such tasks demand very tedious programming efforts from a human programmer. This disadvantage stems from the fact that the present computer is entirely lacking a learning capability which is based upon its own experiences. Hence the human programmer must specify every detail of his instruction to the computer without any single error. Of course it is possible to make the computer look as if it has a learning capability by proper programming but again such programming will be very troublesome for the human programmer. On the other hand, if a learning capability could be incorporated in a computer-like machine, the machine would be able to solve non-arithmetical problems quite efficiently and it could be called a "real" artificial brain. A first step toward such a learning machine is the study of a learning network, namely, a logical network with a learning capability.

A learning network is defined as a variable-parameter network with n binary inputs and m binary outputs, which, given a truth table of n inputs and m "desired" outputs, can by itself adjust its internal parameters so as to eventually produce the desired outputs for any input combinations after the truth table has repeatedly been presented to the

network. A generalized network to be considered in this thesis is shown in Figure 1. It consists of three parts, namely, a variable-parameter circuit, a control circuit and a comparator circuit. When a set of inputs (± 1) and the corresponding desired outputs (± 1) are presented to the network, the variable-parameter circuit produces outputs based upon its existing parameters. The comparator circuit then compares the outputs with the desired outputs. If there is no discrepancy among them, no change occurs and the network is ready to accept the next set of inputs. If, however, they do not agree, Error indication signals (E) are sent from the comparator circuit to the control circuit. This portion of the network constitutes a feedback path. Upon receiving E-signals, the control circuit determines which parameters should be adjusted using Reference signals (R) and sends out Adaptation signals (A). The parameters which received A-signals are varied in such a manner that the outputs become equal to the desired outputs. This process is repeated for the other sets of inputs. Eventually when for all sets of inputs the desired outputs are obtained as the actual outputs, a completion signal (C) is dispatched and the whole process is complete. Now the network is ready to serve as the prescribed function. The iterative process of parameter adjustment may be called a learning process.

The central part of the learning network is the variable-parameter circuit. It has many properties not found in conventional digital circuits. Since it contains the internal parameters as a kind of distributed memory, it is a sequential circuit rather than a combinational circuit. Also it is a hybrid system of digital and analog circuits, because the parameter is an analog quantity. The values of the parameters will be



A: Adaptation signals
R: Reference signals
E: Error indication signals
C: Completion signal

Figure 1. Generalized learning network

the functions of the adaptation currents or voltages not only of the present but also of the past.

As the basic building block of the circuit, a threshold element was selected because of its simplicity and versatility. The threshold element produces a +1 output when the linearly weighted sum of inputs exceeds a certain threshold and otherwise, a -1 output. The weights associated with the inputs and the threshold are variable-parameters. Since they are internally adjustable, the element may be called adaptive.

Historically, the first significant study of the learning network using threshold elements began with the perceptron by F. Rosenblatt (1) in 1957. The perceptron may be regarded essentially as a single threshold element with a very large number (500 to 1000) of inputs which are connected to almost the same number of sensory units through random connections. One of the most important contributions of the perceptron is the proof that there exist certain rules in adjusting the weights which guarantee the convergence of the learning process. Since the number of inputs is large and random connections are involved, a statistical approach was used to investigate the gross behavior of the element. No interest was shown in the macroscopic details of the learning process. In 1961, J. K. Hawkins (2) considered how an arbitrary Boolean function could be realized by learning with a network of cascaded threshold elements and he pointed out some difficulties of the problem. The first learning network which could realize any Boolean functions was shown by R. C. Ridgway III (3) in 1962. In his network, the outputs from several threshold elements in parallel are fed into an OR or a Majority element and its output is taken as the final output. A very important element of Ridgway's work was his development of

the weighted sum criterion as the method of element selection for adaptation. It is noted that there is only a single layer of adaptive elements in his network. A multi-layer learning network where there is more than one layer of adaptive elements has not been studied.

So far only single-output networks were mentioned. Some multi-output networks have also been studied (1,3). But in all cases none of the outputs have any variable parameters in common. A change in a parameter in such networks can affect only one output. All the other outputs are immune to the change. For analysis these multi-output networks can be divided into and considered as single-output networks in parallel. Non-separable multi-output networks are yet to be studied.

In this thesis, some properties of a threshold element will be discussed first in terms of the Hamming distance between input sets. Then the learning process in a single adaptive element will be considered in some detail. As for the networks of threshold elements, attention will be directed to the learning networks which have a small number of inputs and outputs but can realize all possible functions of the inputs. Such networks are called universal. They are quite different from the so-called pattern recognition networks such as the perceptron where the number of inputs may be large (hence a statistical approach is necessary for the analysis) but only a fraction of all possible functions need be recognized. On the other hand, an n-input m-output universal learning network must be able to realize all $(2^{2^n})^m$ functions, which are a tremendous number of functions. For example, a small network of 4 inputs and a single output, or of 3 inputs and 2 outputs should be able to establish 2^{2^4} or $(2^{2^3})^2 = 2^{16} = 65{,}536$ different functions. In a sense, such a learning

network is equivalent to a group of 65,536 conventional logical circuits. Inevitably such a network tends to be complicated but some complication may be offset by its tremendous versatility.

Since only a fraction of all Boolean functions is realizable with a single threshold element, several elements must be combined together to make a universal network. The central problem is to determine which elements are responsible for an error when the final outputs are incorrect, especially when there is more than one layer of adaptive elements. Some criterion is necessary to establish the validity of the intermediate outputs. In a non-separable multi-output network, a change in a single parameter can affect more than one output and as a consequence there can be a conflict if the change is favorable to one output but it is undesirable for another output. A learning procedure must then be developed which will avoid non-resolvable conflicts.

The analysis and design problems will be considered in terms of theoretical models with the aid of computer simulation. Hardware models will be excluded. Because of the small size of the model networks, however, the models have physical realizability and the results obtained by this study will be useful in checking experimental data from those physical networks. Little attention will be paid to a biological analogy with natural neurons. As some physiologists have been warning, the analogy tends to lead to over-simplification and misunderstanding. Enough factual knowledge of the behavior of natural neurons has not been accumulated to permit meaningful analogies to be drawn.

II. PROPERTIES OF A THRESHOLD ELEMENT

A threshold element with n inputs is defined as an element whose output value z is:

$$z = +1 \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq T$$

$$z = -1 \quad \text{if} \quad \sum_{i=1}^{n} w_i x_i < T$$

where x_i is the i-th input variable and takes the value of +1 or -1, w_i is the weight associated with the input x_i and is a real number, and T is the threshold of the element and is also a real number. A set of n weights and a threshold $(w_1, w_2, \ldots, w_n; T)$ is called the structure of a threshold element and the set specifies an n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$. A threshold function is defined as a Boolean function which is realizable with a single threshold element. The properties of a threshold element can be considered in terms of the properties of a threshold function.

Since the number of Boolean functions is quite large, it is convenient to classify them according to their properties. A class of Boolean functions is defined as a group of such functions that could be made identical to each other by any or all of the following transformations:

a. permutation of the variables

b. complementation of the variables

c. complementation of the function.

For example, a two-variable AND function $f_1(x_1, x_2) = x_1 \cdot x_2$ and a two-variable OR function $f_2(x_1, x_2) = x_1 + x_2$ belong to the same class, since

$$\overline{f_2(\bar{x}_1, \bar{x}_2)} = \overline{\bar{x}_1 + \bar{x}_2} = x_1 \cdot x_2 = f_1(x_1, x_2).$$

The following properties of threshold functions will be used in the

discussions to follow. The details are referred to in the paper by S. Muroga (4) and others.

(i) If a Boolean function is a threshold function, then all the functions belonging to the same class as that function are also threshold functions. It is useful to define a representative function from each class of threshold functions as well as from each class of Boolean functions.

(ii) All classes of Boolean functions are not threshold functions. Actually the ratio of the number of threshold functions to the number of all Boolean functions becomes very much smaller as the number n of inputs increases, as seen from Table 1.

Table 1. The ratio of the number of threshold functions (and their classes) to the number of all Boolean functions (and their classes)

n	1	2	3	4	5	6
Functions	$\frac{2}{2}$	$\frac{8}{10}$	$\frac{72}{218}$	$\frac{1536}{64594}$	$\frac{86080}{4\times10^{9}}$	$\frac{1\times10^{7}}{1\times10^{19}}$
(%)	(100)	(80)	(33)	(2.4)	(2×10^{-3})	(10^{-10})
Classes	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{3}{10}$	$\frac{9}{208}$	$\frac{48}{6\times10^{5}}$	$\frac{504}{2\times10^{14}}$

The number does not include those functions having redundant variables whose removal does not affect the values of the functions. The number in parenthesis is the percentage ratio. The table also shows the ratio of the number of classes of threshold functions to the number of classes of Boolean functions. By discounting functions having redundant variables, the latter number of Boolean classes was obtained from Harrison's table (5).

(iii) The set of weights and a threshold which realize a threshold function is not unique. To introduce uniqueness, the "optimum" structure

of a threshold element is defined as a set of such weights and a threshold that makes the sum of the absolute values of weights and a threshold minimum, or

$$\sum_{i=1}^{n} |w_i| + |T| = \text{minimum}$$

The optimum structures of all classes of threshold functions of up to 6 variables have been determined by S. Muroga (4) and others using a linear programming technique.

For compact notation and easy visualization, a vector representation is useful. Let W be the weight vector and X be the input vector. They are defined as:

$$W = (w_0, w_1, w_2, \ldots, w_n) \text{ where } w_0 = -T$$

$$X = (x_0, x_1, x_2, \ldots, x_n) \text{ where } x_0 = +1 \quad .$$

Then by definition, the output z of a threshold element or the value f(X) of a threshold function is:

$$z = f(X) = +1 \quad \text{if} \quad W \cdot X \geq 0$$

$$z = f(X) = -1 \quad \text{if} \quad W \cdot X < 0$$

The following geometrical interpretation is due to W. C. Ridgway III (3).

(a) All sets of weights including the threshold generate a n+1 dimensional space called the weight space. The vectors W and X are vectors in this space.

(b) The equation $W \cdot X = 0$ represents a plane called an input plane. The vector X is normal to the plane. All input planes pass through the origin of the coordinate system describing the weight space.

(c) One side of an input plane $W \cdot X=0$ can be considered to represent a +1 output for the input X and the other side a -1 output.

(d) Since there are 2^n different input planes, the weight space is divided by those 2^n planes (each corresponds to each of 2^n input combinations) into disjoint subspaces.

(e) Each subspace corresponds to a threshold function. Any set of weights in a subspace can be used to represent the threshold function. Thus the set is not unique.

As an illustrative example, consider all Boolean functions of two variables (n=2). Out of 16 possible functions, 14 functions are threshold functions. Let x_1 and x_2 be two input variables. The equation $W \cdot X=w_1x_1+w_2x_2+w_0=0$ represents input planes in the 3-dimensional weight space (w_0,w_1,w_2). Let $w_0=t$, $w_1=u$ and $w_2=v$. The 4 input planes are described by the equations:

$-u-v+t=0$ for $(x_1,x_2)=(-1,-1)$

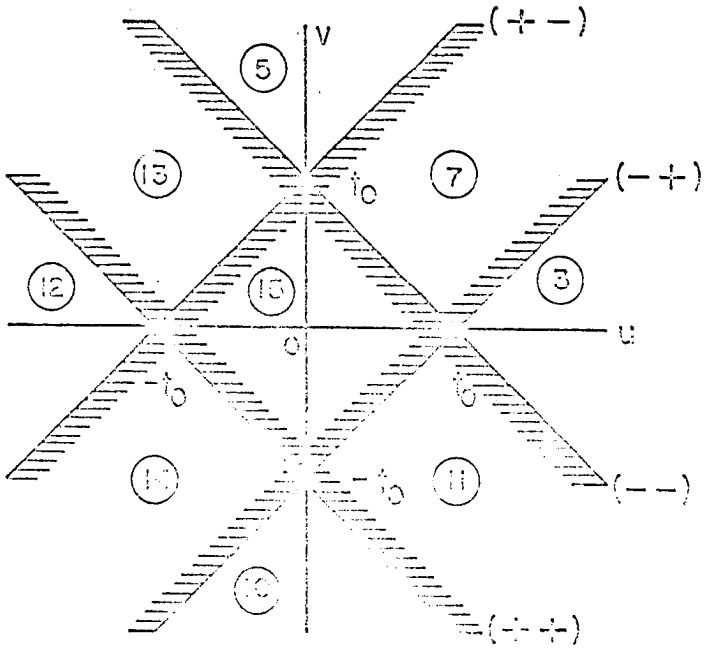
$-u+v+t=0$ for $(x_1,x_2)=(-1,+1)$

$u-v+t=0$ for $(x_1,x_2)=(+1,-1)$

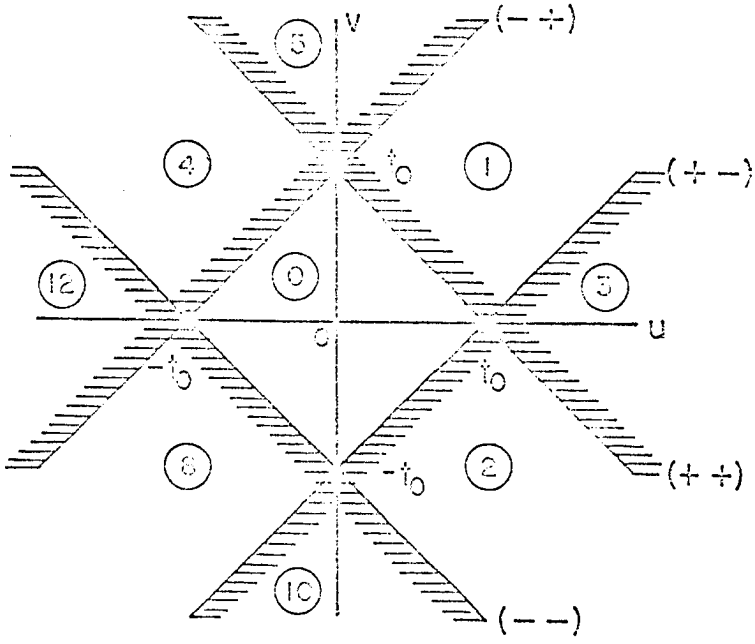
$u+v+t=0$ for $(x_1,x_2)=(+1,+1)$

The 14 subspaces made by these 4 planes are shown in Figure 2. The shaded side of each plane represents a +1 output and the other side a -1 output for the corresponding inputs. For example, the output is +1 for (-1,+1), (+1,-1), (+1,+1) inputs but it is -1 for (-1,-1) input in the subspace (7). The label of each subspace corresponds to one of the following functions:

Figure 2. Subspaces corresponding to 14 two-variable threshold functions



(a) CROSS SECTION AT $t = t_o$



(b) CROSS SECTION AT $t = -t_o$

Labels	0	1	2	3	4	5	7	8	10	11	12	13	14	15	6*	9*
x_1x_2	(0)	(AND)		(x_1)		(x_2)	(OR)		($\bar{x}_2$)		($\bar{x}_1$)			(1)		
- -	-	-	-	-	-	-	-	+	+	+	+	+	+	+	-	+
- +	-	-	-	-	+	+	+	-	-	-	+	+	+	+	+	-
+ -	-	-	+	+	-	-	+	-	+	+	-	-	+	+	+	-
+ +	-	+	-	+	-	+	+	-	-	+	-	+	-	+	-	+

Only the signs of binary values are shown. The names of some functions are written in the parentheses. The functions (6*) and (9*) are not threshold functions and there are no subspaces corresponding to these two functions.

In the rest of this chapter it will be shown that the Hamming distance defined between input terms can be used to see whether or not a Boolean function is a threshold function. So far the value of each input variable x_i has been assumed to be either +1 or -1 instead of 0 and 1, to distinguish a state where x_i does not exist from a state where x_i is zero. But it is sometimes convenient to use 0 and 1. A new variable y_i will be used for this purpose. The conversion is $y_i = (1-x_i)/2$, or $y_i = 0$ when $x_i = +1$ and $y_i = 1$ when $x_i = -1$. This is contrary to the normal convention of taking $y_i = 0$ for $x_i = -1$ but the above conversion is more convenient for reference to other tables as will be shown later. A threshold function is still a threshold function under this conversion (4).

Let $Y = (y_1, y_2, \ldots, y_n)$ be called an <u>input term</u>. The component $y_0 = 0$ corresponding to the threshold is excluded since it is not an input. For brevity the notation $f(Y) = \pm 1$ may be used interchangeably with $f(X) = \pm 1$ without fear of confusion. Y is called a <u>true term</u> of a Boolean function if $f(Y) = +1$. Any Boolean function can be specified by listing all its true terms, each expressed by an integer I, where I is:

$$I = y_n + y_{n-1}2 + y_{n-2}2^2 + \ldots + y_1 2^{n-1} \quad (I = 0,1,2,\ldots,2^n-1) \quad .$$

For example, if f(Y) = +1 only for Y = (0,0,0,0), (0,0,0,1) and (0,0,1,1), then f(Y) may be written as f(Y) = f(0,1,3).

Now the Hamming distance between two input terms Y_1 and Y_2 is defined as:

$$d(Y_1,Y_2) = \sum_{i=1}^{n} |y_{1i}-y_{2i}| = \text{the number of different components of the two vectors } Y_1 \text{ and } Y_2 \ .$$

An input term Y_j is called an isolated true term if $d(Y_j,Y_k) \geq 2$ for all the true terms Y_k other than Y_j.

Theorem 1: The Hamming distance between any two input terms is invariant under; 1) permutation of the input variables, or 2) complementation of the input variables.

Proof: 1) The summation in $d(Y_1,Y_2) = \sum_{i=1}^{n} |y_{1i} - y_{2i}|$ does not depend on the ordering of i. Hence it is invariant for the permutation of the variables. 2) When the i-th variable is complemented, the term $|y_{1i}-y_{2i}|$ becomes $|\bar{y}_{1i}-\bar{y}_{2i}|$. But both cases are 1 when $y_{1i} \neq y_{2i}$ and both are 0 when $y_{1i}=y_{2i}$. Hence $|y_{1i}-y_{2i}| = |\bar{y}_{1i}-\bar{y}_{2i}|$ in either case. Thus the Hamming distance is invariant for the complementation of the variables. Q.E.D.

The input terms Y can be grouped into n+1 sets according to the number of 1's in each Y. Let A(m) = {Y} be the set of input terms where m components of Y are 1. For example, there are 5 sets in the case of n=4.

A(0)	A(1)	A(2)	A(3)	A(4)
(0000)	(0001)	(0011),(1001)	(0111)	(1111)
	(0010)	(0101),(1010)	(1011)	
	(0100)	(0110),(1100)	(1101)	
	(1000)		(1110)	

Theorem 2: (a) For Y_1 and Y_2 ε A(m),

$d(Y_1,Y_2) = 2p$, or an even integer.

(b) For Y_1 ε A(m) and Y_2 ε A(m+1),

$d(Y_1,Y_2) = 2p+1$, or an odd integer.

Proof: (a) Consider only those components which are different between Y_1 and Y_2. If there are p components which are 0 in Y_1 but 1 in Y_2, then there must also be p components which are 1 in Y_1 but 0 in Y_2, becuase the number of 1's must be the same in Y_1 and Y_2. Thus the number of different components is p+p = 2p and hence $d(Y_1,Y_2) = 2p$.

(b) Suppose that there are p components which are 1 in Y_1 but are 0 in Y_2. Then there must be p+1 components which are 0 in Y_1 but are 1 in Y_2, because the number of 1's in Y_2 must be greater than in Y_1 by one. Thus the number of components which are different between Y_1 and Y_2 is p+(p+1) = 2p+1 and hence $d(Y_1,Y_2) = 2p+1$. Q.E.D.

Now with the aid of the above theorems, the following theorem is obtained, which will help to determine whether or not a Boolean function is a threshold function.

Theorem 3: (a) All Boolean functions which have only one true term or 2^n-1 true terms are threshold functions.

(b) A necessary condition for a Boolean function having more than one true term to be a threshold function is that there be no isolated true term in the function.

(c) The above condition is not a sufficient condition.

Proof: (a) Consider a Boolean function f(X) whose value is +1 for only one input term X = (+1,+1,...,+1). This function can be realized with a single threshold element by taking $w_1=w_2=\ldots=w_n=1$ and $w_0= -T=n$, because

$$W \cdot X = \sum_{i=0}^{n} w_i x_i = \sum_{i=1}^{n} x_i - n = 0 \quad \text{for } X = (+1,+1,\ldots,+1)$$

$$W \cdot X = \sum_{i=0}^{n} w_i x_i = \sum_{i=1}^{n} x_i - n < 0 \quad \text{for } X \neq (+1,+1,\ldots,+1).$$

Since all Boolean functions having only one true term and their complements which have 2^n-1 true terms constitute a Boolean class of which the above function is a member, they are all threshold functions.

(b) One can assume that all weights of a threshold function are non-negative without loss of generality, because, if some weights are negative, they can be made positive by complementing the variables associated with those negative weights and the Hamming distance is invariant under the variable complementation by Theorem 1. An isolated true term remains isolated as long as the distance is invariant. Therefore it is sufficient to show that a threshold function with non-negative weights can not have an isolated true term.

Now consider two input terms $Y_1 \in A(m)$ and $Y_2 \in A(m+1)$ such that $d(Y_1,Y_2) = 1$. The proof process of Theorem 2 shows that there is only one component which is 0 in Y_1 and is 1 in Y_2 for p=0. Let it be the j-th component. Then $y_{1j} \neq y_{2j}$. All the other components are equal, namely, $y_{1i} = y_{2i}$, $i \neq j$. Since each weight is assumed to be non-negative, namely, $w_i \geq 0$ for i=1,2,...,n, one obtains the relation:

$$w_0 + \sum_{i=1}^{n} w_i y_{1i} \leq w_0 + \sum_{i=1}^{n} w_i y_{2i}$$

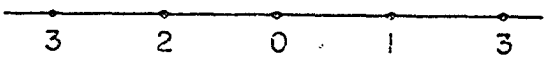
for the above Y_1 and Y_2. This is equivalent to the relation $f(Y_1) \leq f(Y_2)$. This relation holds for any m.

Next suppose that there is an isolated true term in the function and

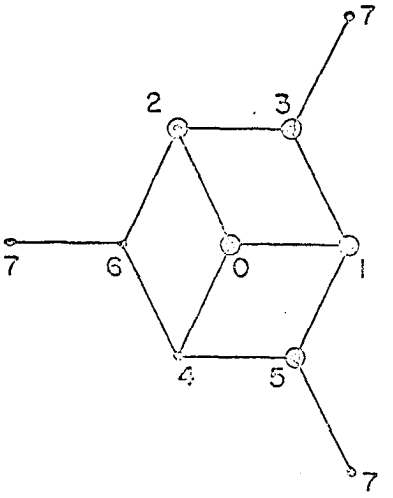
let it be $Y_1 \varepsilon A(m)$. Then $f(Y_1) = 1$. Since Y_1 is isolated, none of $Y_2 \varepsilon A(m+1)$ such that $d(Y_1,Y_2) = 1$ can be true terms, namely, $f(Y_2) = 0$ for all Y_2. This leads to $f(Y_1) = 1 > 0 = f(Y_2)$. Since this contradicts the previously obtained relation $f(Y_1) \leq f(Y_2)$ which hold for any m, it must be concluded that there is no isolated true term in a threshold function with non-negative weights.

(c) A counter-example will suffice to show that the condition is not sufficient. It is known from Minnick's table (6) that a 3-variable function f(0,1,2,5) is not a threshold function, but none of the true terms 0=(000), 1=(001), 2=(010), 5=(101) is isolated. Q.E.D.

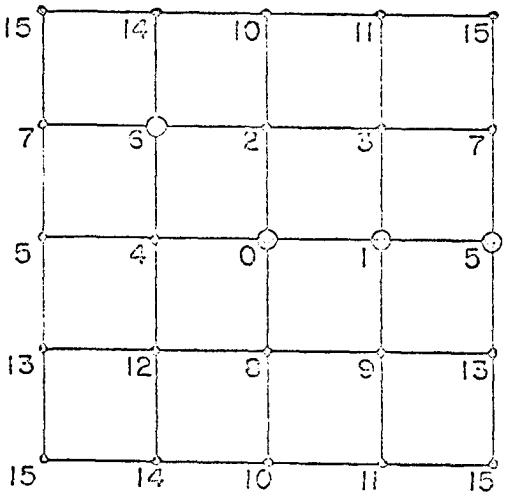
To use the theorem it is convenient to plot 2^n input terms as points according to Hamming distances among them as shown in Figure 3. The plot is made so that each point has n neighboring points at the Hamming distance of 1 in a symmetric position. If the true terms of a Boolean function are marked with heavy dots, it will be easy to see if there is an isolated true term. For example, it will be seen from Figure 3(b) that f(0,1,2,3,5) might be a threshold function, since there is no isolated true term. From Figure 3(c) one can see that f(0,1,5,6) is definitely not a threshold function, because the true term 6 is isolated.



(a) n = 2
(4 TERMS)



(b) n = 3
(8 TERMS)



(c) n = 4
(16 TERMS)

Figure 3. Plot of 2^n input terms

III. LEARNING IN A SINGLE THRESHOLD ELEMENT

A. Learning Procedure

An n-input single-output variable-parameter network consisting of a single threshold element is considered in this chapter. The element is shown in Figure 4. When a set of n inputs and the desired output are given, it is quite easy to adjust the weights (including the threshold) so that the output becomes equal to the desired output. But this set of weights may not be adequate for another set of inputs and the weights may have to be changed again. This in turn may result that the changed weights no longer produce the desired output for the former or other sets of inputs. For a complete learning there must be a guarantee that the iterative process of weight changes makes the weights converge in a finite length sequence into a set of weights which produces the desired outputs for all sets of inputs.

A learning procedure with such guarantee of convergence was used by F. Rosenblatt (1) in the perceptron system and later by others (2,3). A similar procedure will be adopted in this paper. Since a single threshold element is to be used, only threshold functions will be considered in this chapter.

Now define z to be the output of a threshold element and z* to be the desired output for an input $X = (x_0, x_1, \ldots, x_n)$.

Learning procedure (i) for a single element

1. Initially the values of weights w_i are arbitrary.

2. Each of 2^n possible input combinations X and the corresponding desired output z* is presented in an arbitrary (ordered or random) sequence

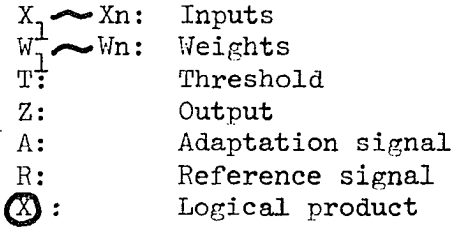
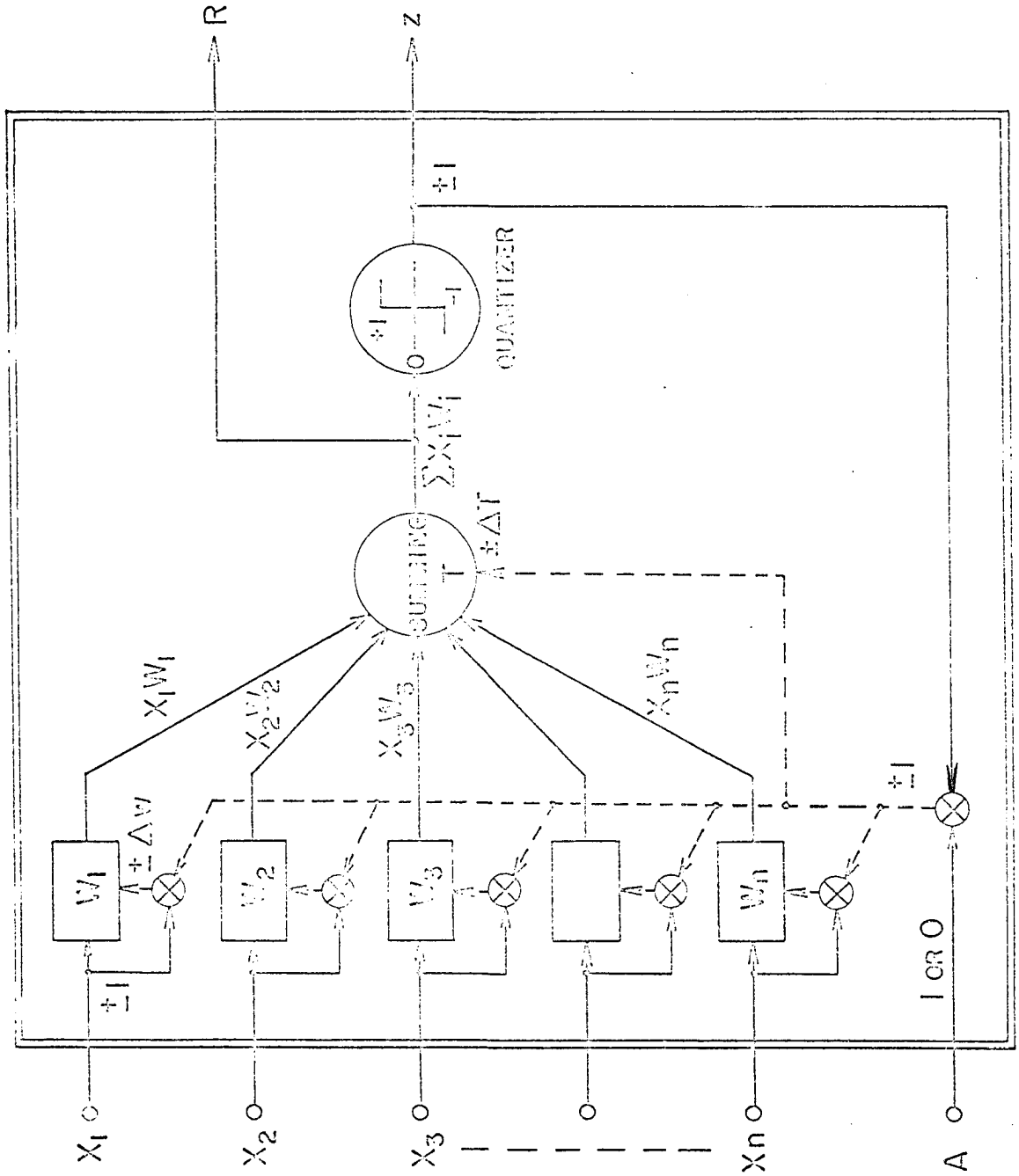


Figure 4. Adaptive threshold element


R
z
±1
QUANTIZER
+1
-1
0
ΣXiWi
T
A ± ΔT
X1W1
X2W2
X3W3
XnWn
W1
W2
W3
Wn
± ΔW
±1
±1
1 OR 0
X1
X2
X3
Xn
A

as many times as necessary for convergence.

3. Whenever $z=z^*$ for an X, the network proceeds to receive the next input combination. No weight change occurs.

4. When $z \neq z^*$, all the weights are changed by the amount ΔW, where $\Delta W = z^* X$ or $\Delta w_i = z^* x_i$, $i=0,1,2,\ldots,n$. The weight change is repeated until $z=z^*$ is obtained. Then the network is ready to receive the next input combination.

5. When $z=z^*$ is obtained for all possible X's, the learning process is complete and no further change occurs.

A proof of convergence for this process is discussed by Rosenblatt (1) and others (2,3). The most lucid exposition may be the proof from a geometrical point of view by W. C. Ridgway III (3). It is shown that:

(a) In the n+1 dimensional weight space, there is a subspace where any point or a set of weights in the subspace can realize a given threshold function. An example has been shown in Chapter II.

(b) The weight point or vector W moves from one side of an input plane (for which it is adapted) to the other side along a normal to the plane, because $\Delta W = z^* X = \pm X$ and the input vector X is normal to the plane as noted in Chapter II.

(c) As a result of a weight change, the distance between the weight point and a certain ideal weight point in the subspace is reduced. This will be seen from the fact that the distance is smaller after adaptation when the weight point is in the same side of the plane as the ideal point than before adaptation when the two points are in the opposite sides of the plane, as long as the point moves normal to the plane and the ideal point is not too close to the plane.

(d) The distance decreases until the actual weight point falls in the subspace of weights that give correct response and at this point the learning is complete.

It is assumed in the above procedure that each weight w_i is changed by a unit amount or $\Delta w_i = z^* x_i = \pm 1$ since $z^* = \pm 1$ and $x_i = \pm 1$. Actually Δw_i may be $\pm d$, where d is a constant, but since w_i is the sum of each such Δw_i, the constant d is merely a scale factor if initially $w_i = 0$. So long as $w_i = 0$ is used as the initial condition, no generality is lost by assuming d=1. Now that d=1, each w_i takes on only integer values. It is not a continuous quantity. However it is also possible to take d arbitrarily small and make the weight function behave as though it were continuous. Henceforth it will be treated as a continuous quantity.

Let the entire step in step 4 be called an <u>adaptation</u>. The following notation is useful in describing the learning process.

s; the number of adaptations

t; the number of weight changes. Since more than one weight change may be necessary during an adaptation, t is greater than s in general, or $t \geq s$

n; the number of inputs

$X(s)=(x_0(s),x_1(s),\ldots,x_n(s))$; the input vector at s-th adaptation. The set $X(1),X(2),\ldots$ does not include such X's for which adaptation does not occur.

$z^*(s)$; the desired output for X(s).

$W(t)=(w_0(t),w_1(t),\ldots,w_n(t))$; the weight vector after t-th weight change. W(t-1) is the weight vector before t-th weight change. W(0) is the initial value of W(t).

$L(t)=|W(t)| = \sqrt{W(t)\cdot W(t)} = \sqrt{\sum_{i=0}^{n} w_i^2(t)}$; the length of a weight vector after t-th weight change.

$\Delta W(s)=z^*(s)X(s)$; the correction vector during s-th adaptation.

$R(s,t)=W(t)\cdot X(s)$; the reference level (or signal) for X(s) after t-th weight change.

$z(s,t)$; the output for X(s) after t-th weight change.

$z(s,t)=+1$ when $R(s,t) \geq 0$ and $z(s,t)=-1$ when $R(s,t) < 0$.

Now before t-th weight change takes place for X(s), the following condition must exist.

$$z(t-1)=-1 \quad \text{or} \quad W(t-1)\cdot X(s) < 0 \quad \text{when } z^*(s)=+1 \tag{1}$$

$$z(t-1)=+1 \quad \text{or} \quad W(t-1)\cdot X(s) \geq 0 \quad \text{when } z^*(s)=-1 \tag{2}$$

In either case,

$$z^*(s)W(t-1)\cdot X(s) \leq 0 \quad . \tag{3}$$

After t-th weight change,

$$W(t)=W(t-1)+\Delta W(s) \tag{4}$$

where

$$\Delta W(s)=z^*(s)X(s) \tag{5}$$

Next suppose that k weight changes have been made for the same X(s), then $W(t+k-1)=W(t-1)+k\Delta W(s)$

and
$$\begin{aligned} W(t+k-1)\cdot X(s) &= W(t-1)\cdot X(s)+k\Delta W(s)\cdot X(s) \\ &= W(t-1)\cdot X(s)+kz^*(s)|X(s)|^2 \\ &= W(t-1)\cdot X(s)+k(n+1)z^*(s) \end{aligned}$$

Here the relation $|X(s)|^2=X(s)\cdot X(s) = \sum_{i=0}^{n} x_i^2=n+1$ is used.

When $z^*(s)=+1$ and if $k \geq -(W(t-1)\cdot X(s))/(n+1)$, then $W(t+k-1)\cdot X(s) \geq 0$ or $z(t+k-1)=+1$.

When $z^*(s)=-1$ and if $k > (W(t-1)\cdot X(s))/(n+1)$, then $W(t+k-1)\cdot X(s) < 0$ or $z(t+k-1)=-1$.

In either case, $z(t+k-1)=z^*(s)$ is obtained and

$$z^*(s)W(t+k-1)\cdot X(s) \geq 0. \tag{6}$$

Thus the desired output is obtained as a result of k weight changes if k is sufficiently large. Actually k is taken as the minimum integer satisfying the above condition. When learning is complete at t=N after M adaptations, then from Equation 6,

$$z^*(s)W(N)\cdot X(s) \geq 0 \quad \text{for all } X(s),\ s=1,2,3,\ldots,M. \tag{7}$$

Theorem 4: 1) The reference level $R(s,t)$ increases or decreases by the amount n+1 as a result of a weight change. Let $\Delta R(s,t)=W(t)\cdot X(s)-W(t-1)\cdot X(s)$, then

$$\Delta R(s,t)=\pm(n+1) \qquad (t \geq s)$$

2) The magnitude of $R(s,t)$ does not exceed n+1, or

$$-(n+1) \leq R(s,t) < n+1 \qquad (t \geq s)$$

Proof: 1) Since $\Delta R(s,t)=W(t)\cdot X(s)-W(t-1)\cdot X(s)=\Delta W(s)\cdot X(s)$, it follows from Equation 5 that $\Delta R(s,t)=z^*(s)|X(s)|^2=(n+1)z^*(s)=\pm(n+1)$.

2) Multiplying both sides of Equation 4 by X(s), one obtains

$$W(t)\cdot X(s)=W(t-1)\cdot X(s)+z^*(s)|X(s)|^2$$
$$=W(t-1)\cdot X(s)+(n+1)z^*(s).$$

When $z^*(s)=+1$, then $W(t-1)\cdot X(s) < 0$ by Equation 1 and

$$R(s,t)=W(t)\cdot X(s) < n+1$$

When $z^*(s)=-1$, then $W(t-1)\cdot X(s) \geq 0$ by Equation 2 and

$$R(s,t)=W(t)\cdot X(s) \geq -(n+1).$$

Putting the two cases together gives

$$-(n+1) \leq R(s,t) < n+1$$ Q.E.D.

Theorem 5: If initially $L(0)=0$, then

$$L(t)=|W(t)| \leq \sqrt{n+1}\,\sqrt{t}$$

Further, if learning is complete at $t=N$, then

$$\sqrt{a(N)}\,\sqrt{N} < L(N) \leq \sqrt{n+1}\,\sqrt{N},$$

where $a(N)=\min_s z^*(s)X(s)\cdot W(N)$.

Proof: Consider the difference $|W(t)|^2 - |W(t-1)|^2$.

If Equation 4 is substituted into $W(t)$,

$$\begin{aligned}|W(t)|^2 - |W(t-1)|^2 &= |W(t-1)+\Delta W(s)|^2 - |W(t-1)|^2 \\ &= 2W(t-1)\cdot\Delta W(s)+|\Delta W(s)|^2 \\ &= 2z^*(s)W(t-1)\cdot X(s)+|\Delta W(s)|^2\end{aligned}$$

The first term is non-positive by Equation 3 and $|\Delta W(s)|^2 = |z^*(s)|^2|X(s)|^2$ $=n+1$. Therefore $|W(t)|^2 - |W(t-1)|^2 \leq n+1$

The summation in both sides over t from 1 to t yields

$$L(t)^2-L(0)^2 = |W(t)|^2 - |W(0)|^2 \leq (n+1)t$$

Since $L(0) = 0$, $\quad L(t) \leq \sqrt{n+1}\,\sqrt{t}$

Now suppose that $W(N)$ is obtained after M adaptations. Let k_s weight changes be made during the s-th adaptation, $s=1,2,\ldots,M$. Then

$$W(N) = \sum_{s=1}^{M} k_s \Delta W(s) = \sum_{s=1}^{M} k_s z^*(s)X(s)$$

and

$$W(N)\cdot W(N)=|W(N)|^2 = L(N)^2 = \sum_{s=1}^{M} k_s z^*(s)X(s)\cdot W(N)$$

Let $a(N)=\min_s z^*(s)X(s)\cdot W(N)=\min_s z^*(s)R(s,N)$. By Equation 7, $a(N) \geq 0$. Since N is the total number of weight changes, it is the sum of the numbers of weight changes made during each adaptation, namely, $N = \sum_{s=1}^{M} k_s$. Then

$$|W(N)|^2 > a(N) \sum_{s=1}^{M} k_s = a(N)N \quad \text{or} \quad L(N) < \sqrt{a(N)}\,\sqrt{N}.$$

Together with the previous result,

$$\sqrt{a(N)}\sqrt{N} < L(N) \leq \sqrt{n+1}\sqrt{N}. \qquad \text{Q.E.D.}$$

B. Computer Simulation

The learning process of a threshold element with up to five inputs was simulated on the Cyclone Digital Computer with the purpose of a further analysis and a comparison with the theory. The simulation program was written under the assumptions:

1. The truth table, a set of 2^n input combinations and the corresponding desired outputs, of a threshold function is repeatedly given.

2. Each time it is given, the inputs are presented in the same fixed order; $0,1,2,3,\ldots,2^n-1$. Thus when the truth table has been given m times, each input combination has been presented exactly m times.

3. All weights are zero at the start, or $w_i=0$, $i=0,1,2,\ldots,n$.

The flow diagram of the simulation program is shown on the next page. All representative functions of up to five variables were tested and confirmed as being realizable after a reasonable number of weight changes. As Table 1 shows, there are 3 representative functions of three variables, 9 such functions of four variables and 48 functions of five variables. The optimum structures of these functions have been calculated by S. Muroga (4) and others as noted in Chapter II. The structures obtained by learning are compared with the optimum structures in Table 2 for all 9 four-variable functions. Four variables are shown as A,B,C and D. The true terms are listed as integers as defined in the previous chapter. The first number in a structure is the weight for A and the second for B, etc. The last number is the threshold T, which is $-w_0$. The N is the total number of

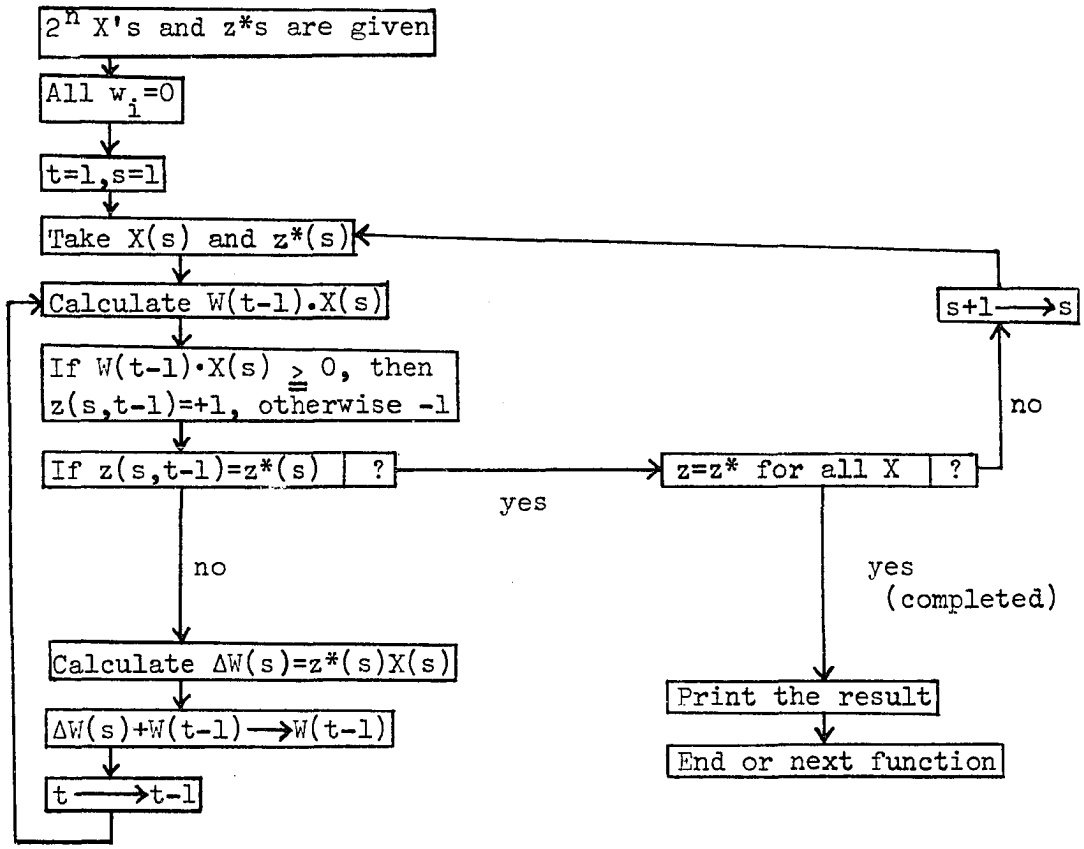


Diagram 1. Flow diagram of the simulation program

weight changes and M is the total number of truth table presentations. For example, 17 weight changes were necessary while the truth table was shown 4 times in order for the element to learn the function f4. Somewhat different figures may have been obtained if a different initial condition and a different input sequence were used.

It is interesting to note that optimum structures were obtained by learning for some functions in spite of the fact that no restriction was imposed on the size of the weight. This implies that at least for these examples the learning procedure has an optimizing tendency.

For 3 threshold functions of three variables which are realizable

Table 2. Comparison of structures produced by learning with optimum structures

Functions	True terms	Optimum structures	Learned structures	N	M
f1=ABCD	0	1 1 1 1;3	the same as optimum	3	1
f2=AB(C+D)	012	2 2 1 1;3	3 3 1 1; 5	13	5
f3=A(BC+CD+DB)	0124	2 1 1 1;2	3 1 1 1; 1	5	2
f4=A(B+CD)	01234	3 2 1 1;2	5 3 1 1; 3	17	4
f5=A(BC+CD+DB)+BCD	01248	1 1 1 1;1	the same as optimum	1	1
f6=A(B+CD)+BCD	012348	2 2 1 1;1	4 4 2 2; 2[a]	12	4
f7=A(B+C+D)	0123456	3 1 1 1;1	the same as optimum	7	2
f8=A(B+C)+BCD	0123458	3 2 2 1;1	5 3 3 1; 1	13	4
f9=A(B+C+D)+BCD	01234568	2 1 1 1;0	4 2 2 2; 0[a]	8	4

[a]Also optimum except for a scale factor

with a 3-input element, 5 weight changes were enough and the truth table was necessary to be shown only once even in the most difficult case.

In the case of 48 five-variable functions which were realized with a 5-input element, the most difficult function required 70 weight changes with 22 presentations of the truth table. Optimum structures were obtained in 20 functions. The maximum weight was 10.

An example of how each weight changes as learning goes on will be seen in Figure 5. Three of six weights (one of which is the threshold) of a 5-variable function f(0,1,2,3,4,5,6,7,8,9,10,16,17) are shown. The function is the most difficult function in the sense that it required the most weight changes. It is very interesting to observe that each weight

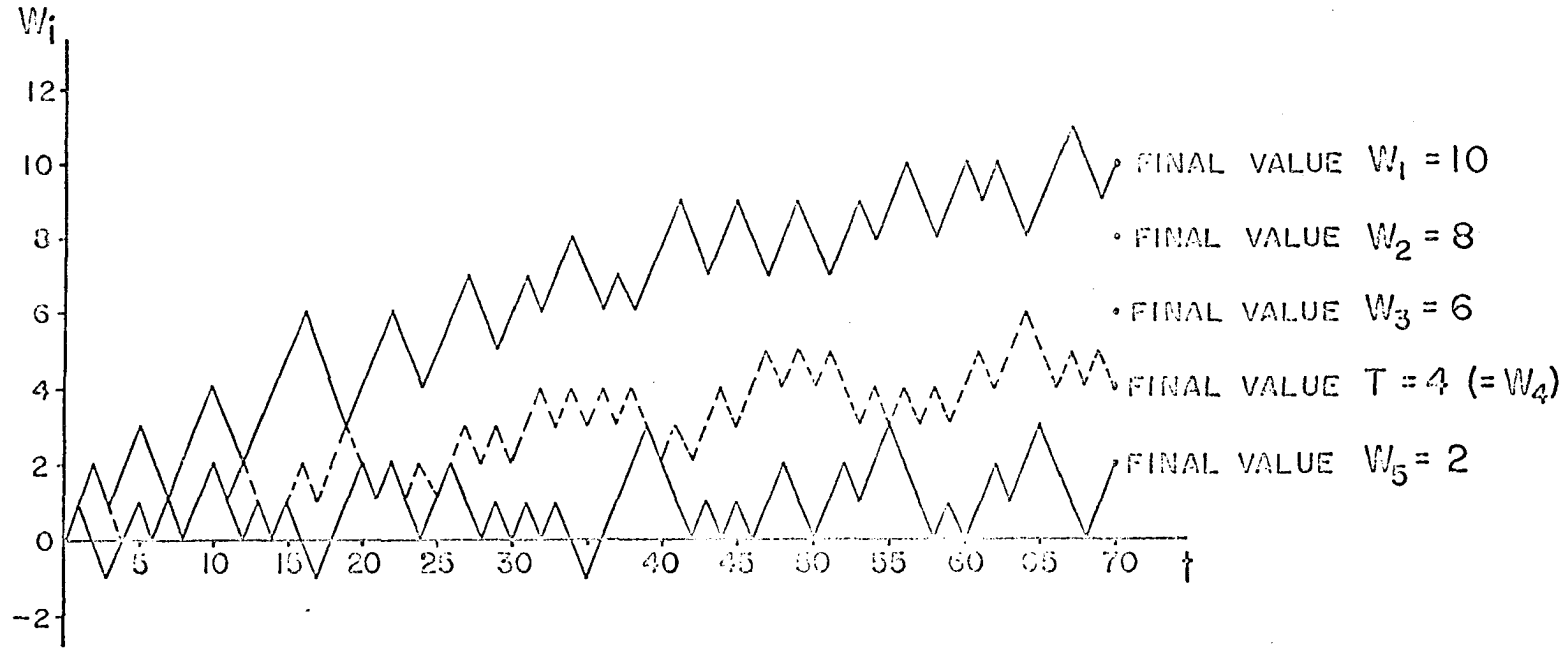


Figure 5. Values of the weights W_1, W_2 and T at each step of weight change for the most difficult 5-variable threshold function

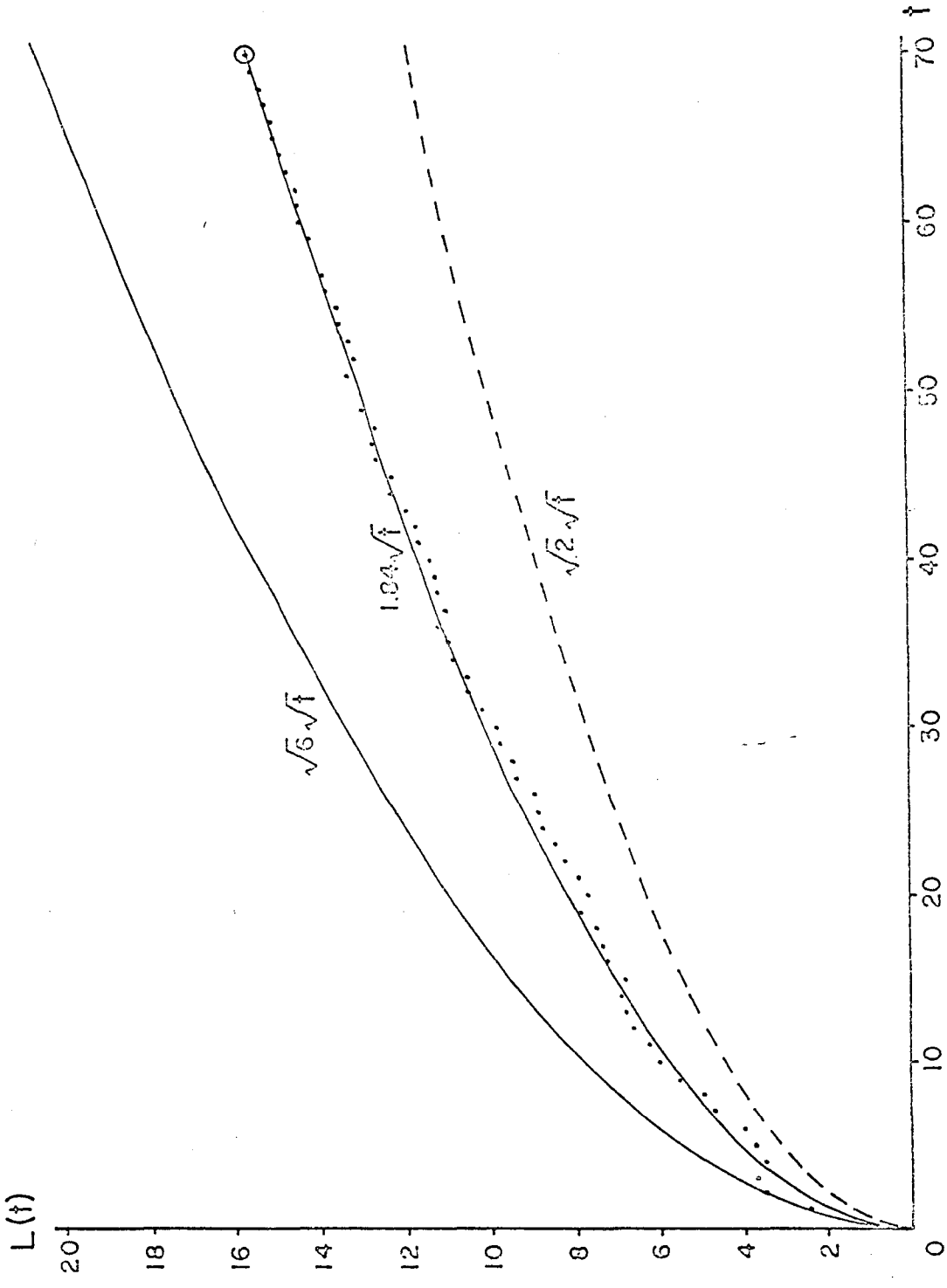
seems to steadily proceed to its goal, the final weight value, in a zigzag way, repeating "increase", "no too much, decrease", "no too much, increase" and so on. In Figure 6, the change of the length of a weight vector for the same function is shown along with the theoretical upper bound obtained from Theorem 5. Since n=5, the upper bound is $\sqrt{n+1}\sqrt{t}=\sqrt{6}\sqrt{t}=2.47\sqrt{t}$. As it turned out, $L(t)=1.84\sqrt{t}$ is a very good approximation in this case. The second part of the same theorem gives $\sqrt{a(N)}\sqrt{N} < L(N) \leq \sqrt{6}\sqrt{N}$. It turned out that $a(N)=\min_S z^*(s)X(s)\cdot W(N)=2$ with $N=70$. $L(N)=\sqrt{2\times 70}$ is shown as a point in the figure. The curve shown by a dotted line was drawn according to $L(t)=\sqrt{2}\sqrt{t}$ as if this were a theoretical lower bound, though it is not except for t=N. On the other hand, from $\sqrt{a(N)}\sqrt{N} < L(N) \leq \sqrt{6}\sqrt{N}$, one can obtain the relation

$$L(N)^2/6 \leq N < L(N)^2/a(N)$$

Then it is possible to plot the final length of weight vectors versus the total number of weight changes required to complete learning and to compare it with the above theoretical limits. Such a plot is shown in Figure 7 for all 48 threshold functions of five variables. The upper limit is shown as $N = \frac{1}{2} L(N)^2$ assuming $a(N)=2$ for all the functions.

So far the initial condition is assumed to be zero for all the weights. But actually one may assume any initial condition. Generally a favorable initial condition quickens the learning process considerably. In particular if the initial values of weights happen to be such that the function is already realizable, then there is no need of adaptation. On the other hand it was found that even an adverse initial condition is rapidly improved during first few adaptations. An example follows. The function

Figure 6. Change of the length of the weight vector of the most difficult 5-variable threshold function


L(t)
20
18
16
14
12
10
8
6
4
2
0
10
20
30
40
50
60
70
t
$\sqrt{6}\sqrt{t}$
$1.84\sqrt{t}$
$\sqrt{2}\sqrt{t}$

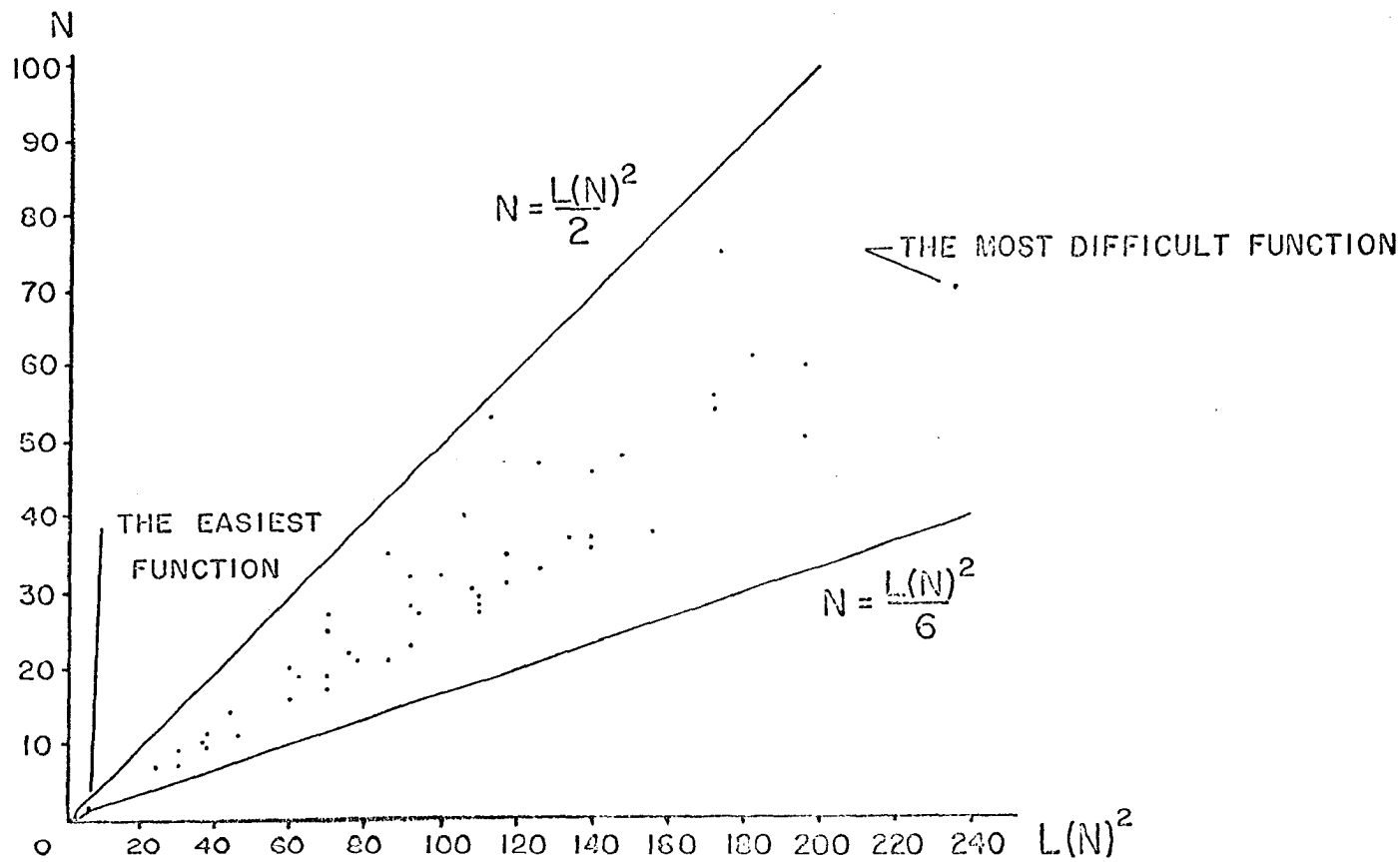


Figure 7. Relation between N and $L(N)^2 (= \sum_{i=0}^{5} W_i^2)$ for 5-variable threshold functions

in question is the 5-variable function termed the most difficult.

(1) Standard initial condition; all w=0.

Learned structure=(10,8,6,4,2;4) N=70, M=18

(2) Favorable initial condition; all w=10.

Learned structure=(15,13,9,7,3;5) N=15, M=2

(3) Adverse initial condition; all w=-10.

Learned structure=(10,8,6,4,2;4) N=82, M=16

Of 82 weight changes in the case 3), 7,4,2 and 5 changes occurred during the first, second, third and fourth adaptations, respectively, and hence less presentations (M) of the truth table were required than in the case 1).

IV. MULTI-ELEMENT UNIVERSAL NETWORKS

A binary network with n inputs and a single output is said to be universal, if any of 2^n possible Boolean functions of n inputs is realizable with the network. As noted earlier, not all Boolean functions are realizable with a single threshold element. But it is possible to build a universal network with a number of threshold elements in cascade. In addition, if an adequate learning procedure is established, the network can become a universal learning network. There are several factors to be considered in building such a network.

1. Whether or not exactly identical elements should be used in all locations.

2. Whether or not fixed weight elements may be used together with adaptive threshold elements.

3. Whether or not the inputs to the network are to be restricted to the first stage or permitted as inputs to later stages.

4. How the interconnection of elements should be made.

The choice of a particular network depends on two generally conflicting factors. For economy a network with fewer elements and with fewer adjustable parameters is desirable. For high learning efficiency, however, some redundancy is indispensable. In this chapter various universal networks with varying degrees of redundancy will be considered and compared.

First, the minimum number of threshold elements required in a universal network is considered. In general the number N(n) of n-variable threshold functions is known to be less than $2^{n^2+1}/n!$ (7), or $N(n) < 2^{n^2+1}/n!$ for n=2,3,4... . When p identical elements, each with

n inputs, are combined, the number of all possible functions realizable with the combination is at most $N(n)^P$. For such a p-element network to be universal, this number must be equal to or greater than 2^{2^n}, or $N(n)^P \geq 2^{2^n}$. Combining the two relations, one obtains $(2^{n^2+1}/n!)^P \geq 2^{2^n}$ or $p \geq 2^n/(n^2+1-\log_2 n!)$. The values of minimum p which satisfies this relation are shown in Table 3 for small n. This lower bound of the number of elements is better than those obtained elsewhere (3,7). As n becomes larger, the lower bound increases very rapidly, indicating that really very many elements are necessary to make a universal network. For example, min. p=60 for n=13 and min. p=308 for n=16.

To get an upper bound of p, one must be able to show that a synthesis is possible with a certain number of elements. An available general synthesis is a network whose output is an OR function of outputs of a number of threshold elements. It can be easily shown (4) that such a network can be built with 2^{n-1} threshold elements connected in parallel to an OR element having 2^{n-1} inputs. Thus the number $2^{n-1}+1$ shown as max. p in Table 3 may be considered as an upper bound of the number of elements necessary for a universal network. Obviously this is one of the most primitive synthesis procedures. The number of elements could be considerably reduced by more efficient synthesis procedures.

Table 3. The minimum and maximum numbers of threshold elements required in a universal network

n	2	3	4	5	6	7	8	9	10
min. p	1	2	2	2	3	4	6	9	13
max. p	3	5	9	17	33	65	129	257	513

A simple way of making a universal network is to employ 2^n elements each with n inputs, assigning each of 2^n input terms to each element and then to make a final output by an OR element from the outputs of those elements. An example of such a 3-input universal network is shown in Figure 8. It consists of 8 identical threshold elements with weights all equal to +1 or -1 and an OR element. When a set of inputs (± 1) is given, the weighted sum of inputs exceeds a common threshold of 3 (shown in a circle) only in one element. Then its output is +1 but all the other outputs are -1. If the input set is a true term, a weight of +1 is assigned to the +1 output but otherwise a weight of 0 is assigned to the +1 output. The network may be called a truth-table-type network, since it is a direct realization of a truth table. If the network is to be a learning network, the learning procedure will simply consist of assigning a +1 or 0 value to each of 8 parameters. This is an example of simplified learning at the cost of a large number of elements. The fact that learning is trivially simple may deserve attention for some applications.

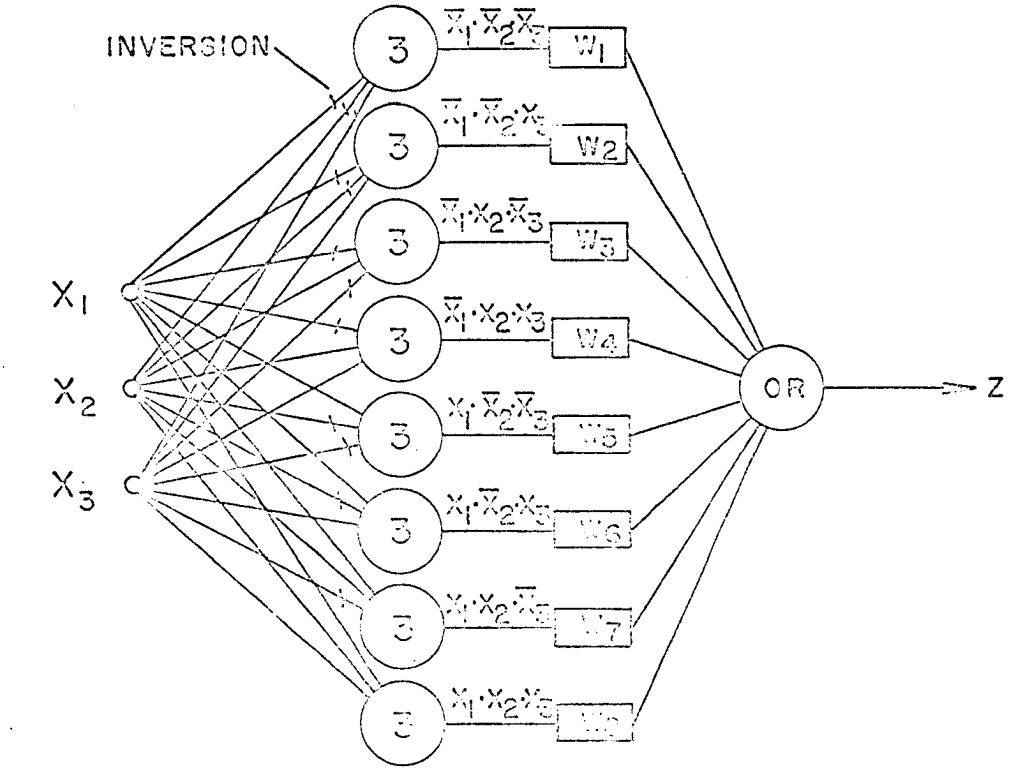
Another example of a 3-input universal network is shown in Figure 9. It consists of 4 AND elements and a threshold element. By assigning appropriate values to the weights and the threshold of the last element, any 3-variable function is realizable with this network. The assignment can be done also by learning in exactly the same way as in a single element. It is interesting to note that the final output z is:

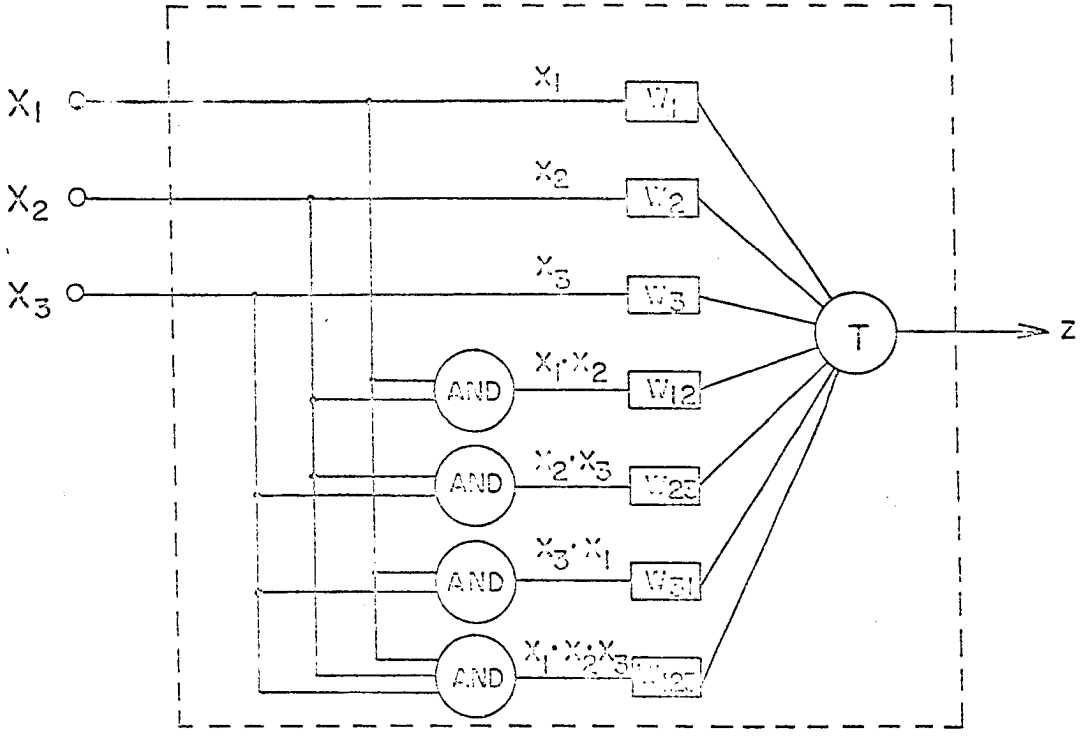
$$z=+1 \quad \text{if} \quad w_1x_1 + w_2x_2 + w_3x_3 + w_{12}x_1x_2 + w_{23}x_2x_3 + w_{31}x_3x_1 + w_{123}x_1x_2x_3 \geq T$$

$$z=-1 \quad \text{if the sum is less than T.}$$

Figure 8. Truth-table-type network
(w_i=0 or +1)

Figure 9. Nonlinear threshold element


INVERSION
3
3
3
3
3
3
3
3
$\bar{X}_1 \cdot \bar{X}_2 \bar{X}_3$
$\bar{X}_1 \cdot \bar{X}_2 X_3$
$\bar{X}_1 \cdot X_2 \bar{X}_3$
$\bar{X}_1 \cdot X_2 X_3$
$X_1 \cdot \bar{X}_2 \bar{X}_3$
$X_1 \cdot \bar{X}_2 X_3$
$X_1 \cdot X_2 \bar{X}_3$
$X_1 \cdot X_2 X_3$
W1
W2
W3
W4
W5
W6
W7
W8
X_1
X_2
X_3
OR
Z


X_1
X_2
X_3
X_1
X_2
X_3
$X_1 \cdot X_2$
$X_2 \cdot X_3$
$X_3 \cdot X_1$
$X_1 \cdot X_2 X_3$
W1
W2
W3
W12
W23
W31
W123
AND
AND
AND
AND
T
Z

Compared with a threshold element, the defining equation has additional "nonlinear" terms. If the entire network is regarded as a single element, it may be called a "nonlinear" threshold element. A generalization leads to the following theorem, where binary values of 0 and 1 are used in place of +1 and -1 for simplicity.

Theorem 6: Any arbitrary Boolean function is realizable with a single nonlinear threshold element whose output z is defined by the following nonlinear equation.

$$z=1 \quad \text{when} \quad w_1y_1 + w_2y_2 + \ldots + w_ny_n + w_{12}y_1y_2 + \ldots + w_{123}y_1y_2y_3 + \ldots + w_{12\text{-}n}y_1y_2\ldots y_n \geq T$$

$z=0$ when the sum is less than T.

Moreover a set of such weights w_i's and a threshold T can be calculated in a straightforward way.

Proof: A Boolean function can be written as a logical sum of true terms with each term being expressed as a logical product of variables or their complements. If the complement of a variable y_i is written as $1-y_i$, a function can be rewritten as an algebraic equation (8). A true term has a form like $\bar{y}_1y_2\ldots\bar{y}_i\ldots y_n$. Substituting y_i by $1-y_i$, one can rewrite the term as $(1-y_1)y_2\ldots(1-y_i)\ldots y_n$. The sum of such terms has the form of the above defining equation and it has a value of +1 for true terms and a value of 0 for other terms, or

$$w_0 + w_1y_1 + \ldots + w_ny_n + w_{12}y_1y_2 + \ldots + w_{123}y_1y_2y_3 + \ldots + w_{12\text{-}n}y_1y_2\ldots y_n = 1$$

for true terms and the sum is 0 for other input terms.

Thus it is always possible to determine a set of w_i's and a T by rewriting the logical sum of true terms with the notation $1-y_i$ for the complement of a variable y_i. Q.E.D.

As an example, consider a 3-variable function

$$f(0,3,5) = \bar{y}_1\bar{y}_2\bar{y}_3 + \bar{y}_1 y_2 y_3 + y_1\bar{y}_2 y_3 \quad .$$

Substituting y_i with $1-y_i$, one obtains:

$$f(0,3,5)=(1-y_1)(1-y_2)(1-y_3)+(1-y_1)y_2y_3+y_1y_3(1-y_2)$$

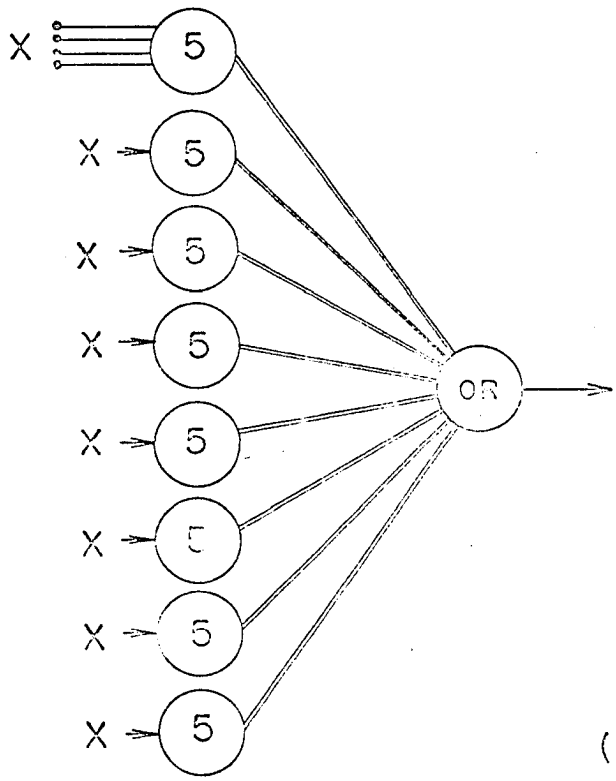
$$=1-y_1-y_2-y_3 + y_1y_2 + 2y_2y_3 + 2y_3y_1 - 3y_1y_2y_3 \quad .$$

Thus $w_0=1$ or $T=1-w_0=0$, $w_1=w_2=w_3=-1$, $w_{12}=1$, $w_{23}=w_{31}=2$ and $w_{123}= -3$.

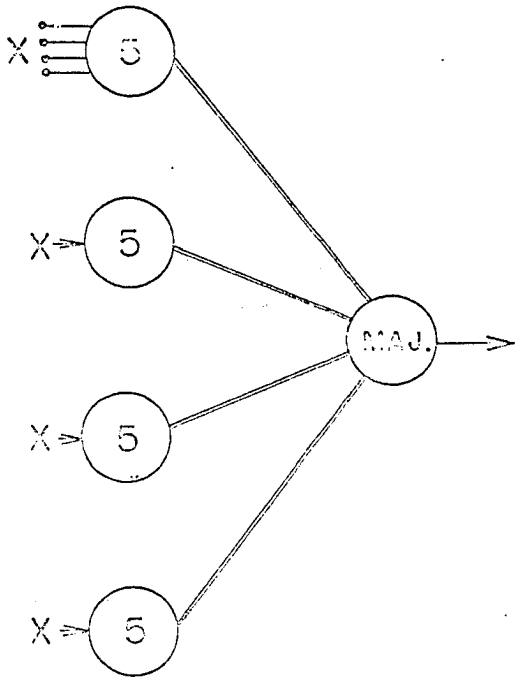
The question why a threshold element is not universal may be answered by the fact that nonlinear terms are missing from the defining equation of a threshold element. In this sense a threshold element should be called a "linear" threshold element. While a linear threshold element has n+1 parameters, a nonlinear threshold element has 2^n parameters. Though a nonlinear threshold element has a complicated structure as an element, the number 2^n of its parameters is small compared with those of other universal networks at least for small n as will be shown later in this chapter. Moreover learning can be simply done as in a single threshold element, the only difference being the number of the parameters. The convergence of the learning process is guaranteed because a set of weights and a threshold is known to exist by Theorem 6. If the construction of a nonlinear threshold element is physically feasible with a reasonable effort, the element will deserve serious attention because of its tremendous versatility and high learning efficiency due to a small number of parameters and simple learning.

A standard way of synthesizing a universal network is to combine a number of elements. Four multi-element networks of threshold elements as shown in Figure 10 are considered in the rest of this chapter and in the

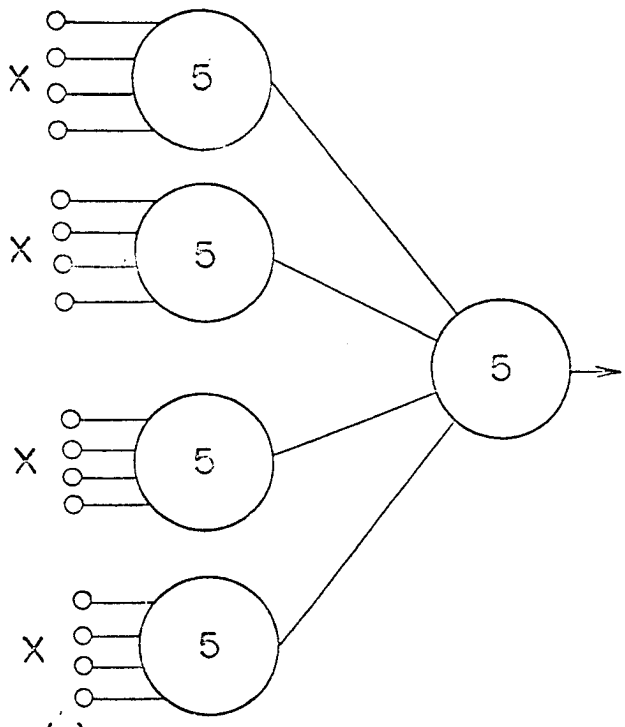
Figure 10. Universal networks of threshold elements (The number in a circle shows the number of parameters)



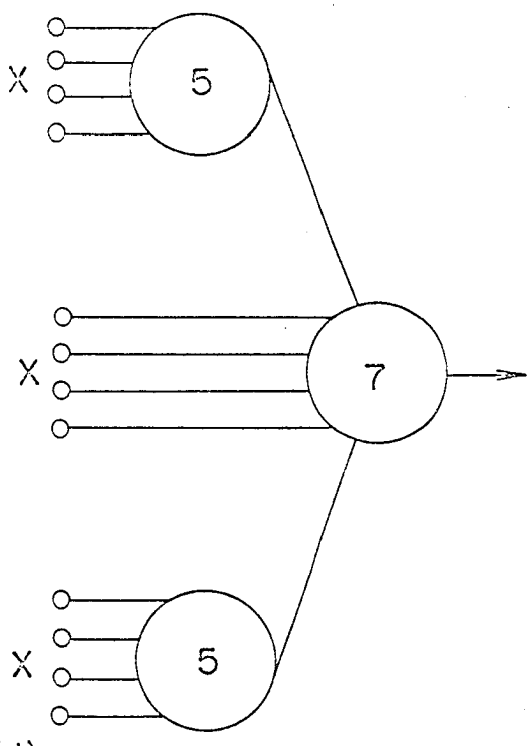
(a) OR NETWORK



(b) MAJORITY NETWORK



(c) SYMMETRIC NETWORK



(d) ASYMMETRIC NETWORK

next chapter. Let the elements in the first stages be called input elements and those in the second stages be called output elements.

(a) OR network.....The output element is an OR element. The configuration is the simplest. The total number of elements is $2^{n-1}+1$ and the number of parameters is $2^{n-1} \times (n+1)$. Both are very large.

(b) Majority network.....The output element is a Majority element whose output is determined by the majority of its inputs. Less elements are required than in (a). If there are p elements in the network, the number of parameters is $(p-1) \times (n+1)$, where p includes the output element.

(c) Symmetric network.....All elements including the output element are identical and adjustable. Thus adaptive elements are distributed in two layers, while in (a) and (b) they are only in a single layer. If there are p elements, the number of parameters is $p(n+1)$.

(d) Asymmetric network.....The output element is also adaptive but it has some extra inputs. Thus all elements are not exactly identical. Moreover it is assumed that all inputs are available not only in the first stage but also in the second stage. Unless the inputs are slow in variation, delay elements may be needed for the inputs to the output element. If the network has p elements in 2 layers, the number of parameters is $p(n+1)+(p-1)$. It is smaller than those of the other three networks at least for small n.

Though Figure 10 shows 4-input networks, the extension to networks with more than 4 inputs would be obvious. In (a) and (b), the networks would be extended only "vertically" as n increases. They would be expanded both "vertically" and "horizontally" in (c) and (d).

Table 4 compares the numbers of parameters of a threshold element,

of a nonlinear threshold element and of the four networks. The number in parenthesis shows the number of elements in a network. The numbers of necessary elements in (b) and (c) are obtained as a result of a computer simulation as will be explained later. The number of necessary elements in (d) is due to R. C. Minnick (6) but it was also confirmed by the computer simulation. The figures for n=5 in (b), (c) and (d) are those estimated and not confirmed. The figures in the last column show the lower bound of the number of necessary elements taken from Table 3. The lower bound of the number of parameters is shown as p(n+1) where p is the lower bound of the number of necessary elements.

Table 4. Comparison of the number of parameters (and the number of elements) in an element and various universal networks with n inputs.

n	Single Element	Nonlinear Element	(a)	(b)	(c)	(d)	Lower bound
1	2	2	2 (1)	2 (1)	2 (1)	2 (1)	2 (1)
2	3	4	6 (3)	6 (3)	9 (3)	7 (2)	6 (2)
3	4	8	16 (5)	12 (4)	16 (4)	9 (2)	8 (2)
4	5	16	40 (9)	20 (5)	25 (5)	17 (3)	10 (2)
5	6	32	96 (17)	30 (6)?	36 (6)?	20 (3)?	12 (2)

V. LEARNING IN UNIVERSAL NETWORKS

A. Learning Procedure

Learning in multi-element networks having n inputs and a single output as shown in Figure 10 is considered in this chapter. Each element has n+1 or more weights including the threshold. As in a single-element network the learning process is an iterative process of changing the weights so that a given function is realized with the network. Arbitrary Boolean functions, not necessarily threshold functions, are considered in this chapter. Since the network contains more than one element, the problem is in deciding in which element the weights should be changed first when there is a need of weight change. A reference level criterion was used to select an element for weight change by W. C. Ridgway III (3) in his OR and Majority networks similar to (a) and (b) in Figure 10. It will be shown that the similar criterion can be used also in the networks (c) and (d) in Figure 10, where adaptive elements are distributed in more than one layer and there are intermediate outputs which constitute inputs to the other adaptive elements.

Suppose that the desired output is $z^*(s) = +1$ for an input $X(s)$ at s-th adaptation but the actual output is $z(s, t-1) = -1$ before t-th weight change. It may be reasonable to assume that the elements producing -1 outputs are responsible for the error. This is obvious in the networks (a) and (b). But even in (c) and (d) the assumption is useful because it gives a directionality to a weight change. Under this assumption, the weights associated with an output element in (c) or (d) tend to become positive. Now that the output is in error, k weight changes should be

made in at least one of such elements so that its output is reversed to become +1. For efficient learning it is desirable to make k as small as possible. If the element which has the smallest reference level with a minus sign (the sign opposite to that of the desired output) or $R(s,t-1)=W(t-1).X(s) < 0$ is taken, then the level will be reversed or $R(s,t+k-1)=W(t+k-1).X(s)=R(s,t-1)+k(n+1) \geq 0$ with the smallest k. Similarly when $z^*(s) = -1$ but $z(s,t-1) = +1$, the element which has zero or the smallest reference level with a plus sign (the sign opposite to that of the desired output) is to be selected. Since the least weight change will be made, its effect will be the least for those input combinations that the network has learned to correctly respond to.

A limiting requirement will be imposed on the number of elements in a learning network. If it is known that a network of p elements can realize a particular Boolean function but any network of p-1 elements can not realize that function, then the requirement is that the function must be realizable by learning in a network of just p elements as well as in a network of more elements. Such a p-element learning network has the minimum redundancy in terms of the number of elements. But it has redundancy in the form of weight values. Generally learning would be easier in a network of more than p-elements than in a network of just p elements because of greater redundancy.

As it turned out, the following learning procedure which employs the reference level criterion is effective for all three types of networks (a), (b) and (d) with the minimum number of necessary elements as considered in the previous chapter.

Learning procedure (ii) for (a), (b) and (d)

1. Initially the values of weights are all zero.

2. Each of 2^n possible input combinations X's and corresponding desired output z* is presented in a random or non-cyclic sequence as many times as necessary for convergence.

3. When the output z is $z=z^*$ for an X, no change occurs and the network takes the next input combination.

4. When $z \neq z^*$, the element which has the smallest reference level with the sign opposite to that of z* (zero is regarded as having a plus sign) is selected, and its weights are changed by the amount ΔW, where $\Delta W = z^* X$ or $\Delta w_i = z^* x_i$, for $i=0,1,2,\ldots,n$. The weight change is repeated until the output of that element is reversed. As a result, if $z=z^*$ is obtained, the network proceeds to take the next inputs. But if still $z \neq z^*$, then the entire step is repeated for the other elements.

5. In step 4 above, if two or more elements have the same smallest reference level, one of them must be selected. It affects the learning process how the selection is made in such a case.

6. When $z=z^*$ is obtained for all X's, the learning process is complete and no further change of weights occurs.

A modification is necessary for a symmetric network like (c), where there is more than one layer of adaptive elements. If the above procedure is followed as it is, convergence cannot always be guaranteed for the following reason. Consider two different inputs X_1 and X_2. Suppose that $z^* = -1$ for X_1 and $z^* = +1$ for X_2 and all intermediate outputs happen to be -1 for both X_1 and X_2. Now if $z = +1$ for X_1, then the weights of the

output element must be changed because all intermediate outputs are -1 and have the same sign as z*. Next if z = -1 for X_2 and the reference level of the output element happens to be the smallest, then again the weights of the output element must be changed. If both sets of weights make correct outputs for all the other input combinations, then the outputs of the input elements become immune to inputs and they will not have a chance to be adapted and the same weight change will be repeated for the output element. This can be avoided by establishing a hierarchy for element selection. The following modified learning procedure is used for the network with multi-layers of adaptive elements like (c).

Learning procedure (iii) for (c)

1. When $z \neq z^*$, one of the input elements which has the smallest reference level with the sign opposite to z* is selected and its weights are changed so that the output of that element is reversed. As a result, if $z=z^*$ is obtained, the network proceeds to receive the next inputs. But if still $z \neq z^*$, then the weights of the output element are modified so that $z=z^*$ is obtained.

2. Other details are the same as in Learning Procedure (ii). In case there are more layers, the change is to be made first in the first layer, next in the second layer, then in the third layer and so on.

The initial condition for weight values was arbitrary in the single element case but it cannot be so in a multi-element network in general. For example, if the threshold of an element in a p-element network is very large in the beginning, its reference level will never be the smallest of all the elements. Its weights including the threshold will never

have a chance to be changed. Thus in effect the network has only p-1 elements which cannot realize the function. In general, a zero initial condition will be the best since the reference level is discriminated at zero. Depending upon the function in question, one may assume other initial conditions.

The input sequence was also arbitrary in a single element network. It may be cyclic as $0,1,2,\ldots,2^n-1$; $0,1,2,\ldots,2^n-1$; $0,\ldots$. But if inputs are given in such a sequence to a multi-element network, it is probable that the same sequence of weight changes covering several elements repeats itself and the learning process never converges. Therefore the inputs must be given in such a sequence that never invokes any cyclic weight changes. Of course, inputs may be given in a cyclic sequence for some functions.

When two or more elements have the same reference level, one of them may be selected at random or in some deterministic way. As will be discussed later, the total number of weight changes and the structure of elements obtained by learning will depend on this selection. This problem, however, will not be so significant in actual electronic circuits, because two elements will never have exactly the same reference level due to the presence of noise and the selection will in effect be made at random. In a computer simulation it must be taken into account.

A rigorous mathematical proof of convergence of these learning processes is not established in this thesis, though it is conjectured that a finite length convergence process exists, if cyclic weight changes do not occur. The requirement of a zero initial condition and a random input sequence is a necessary condition for convergence. One resort of the

proof is to show by a computer simulation that the process does converge. The result of such a computer simulation will be discussed later in this chapter.

Now consider a p-element network and a Boolean function realizable with it. One can imagine p weight vectors associated with p elements in the weight space. One can also imagine that there are p "ideal" weight vectors (though they are not unique) whose combination can realize the function. The combination depends on the configuration of elements in the network. Each ideal weight vector W* lies in a subspace corresponding to a threshold function. As learning goes on, each weight vector W starting from the origin moves gradually toward a subspace where an ideal weight vector lies.

As an example, consider a 2-variable function f(0,3) which is +1 for (+1,+1) and (-1,-1) inputs but -1 for (+1,-1) and (-1,+1) inputs. The function is realizable with a network of two adaptive threshold elements connected to an OR output element. Let two weight vectors be W_1 and W_2. Suppose that the inputs are given in the sequence; 2,1,3,0,1,2,..., where 0=(+1,+1), 1=(+1,-1), 2=(-1,+1) and 3=(-1,-1). The movement of W_1 and W_2 can be considered in a 3-dimensional space as shown in Figure 11, where a projection to a 2-dimensional W_1-W_2 plane was made. The four input planes are schematically shown. It will be seen that W_1 and W_2 coincided with W_1* and W_2* after 6 adaptations. At points a, b and c, W_1 and W_2 had the same reference level as a simple calculation would reveal, and one of them was chosen at random in each case. If a different input sequence; 0,3,..., had been used instead, then learning would have been complete after only two weight changes. In both cases the final structures of the two vectors

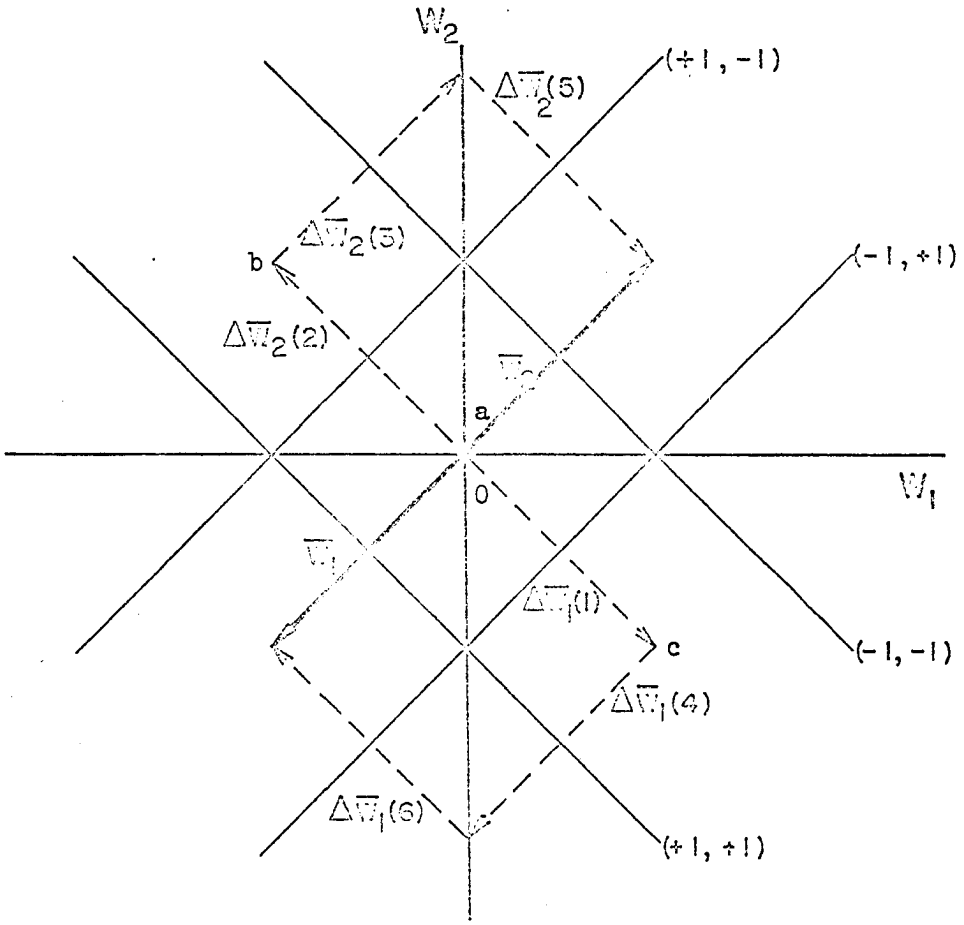


Figure 11. Movement of two weight vectors

are equal and they are $W_1=(w_0,w_1,w_2)=(-1,-1,-1)$ and $W_2=(-1,+1,+1)$. Actually the total number of weight changes as well as the final values of the weights may depend on three factors; 1) initial condition, if a condition other than zero is used, 2) input sequence, and 3) choice of an element, when two or more elements have the same reference level. As the above example suggests, there will be some optimum conditions, or best strategies, of the three factors which make the number of total weight changes minimum.

B. Computer Simulation

To study the convergence property of this learning process, all representative functions of Boolean classes of three and four variables were tested for convergence by a computer simulation.

As seen from Table 1 there are 10 representative functions (one from each class) of 3 variables, of which only 3 are realizable with a single threshold element and the other 7 require more than one element. As for 4-variable functions, there are 208 representative functions, of which only 9 are realizable with a single element and the other 199 require more than one element. When there are 5 or more variables, the number of the representative functions is quite large (of the order of 10^5 for 5 inputs). It requires an unreasonable amount of computer time to simulate the learning process of all of them.

Ordinarily it is not easy to find appropriate weight values which realize a particular function by a computational method. The computer simulation provides a method of determining such weights, though the set of weights might not be optimum in the sense that the absolute sum of

weights should be minimum.

The flow diagram of the program used for the network (d) is shown in Diagram 2 as an example.

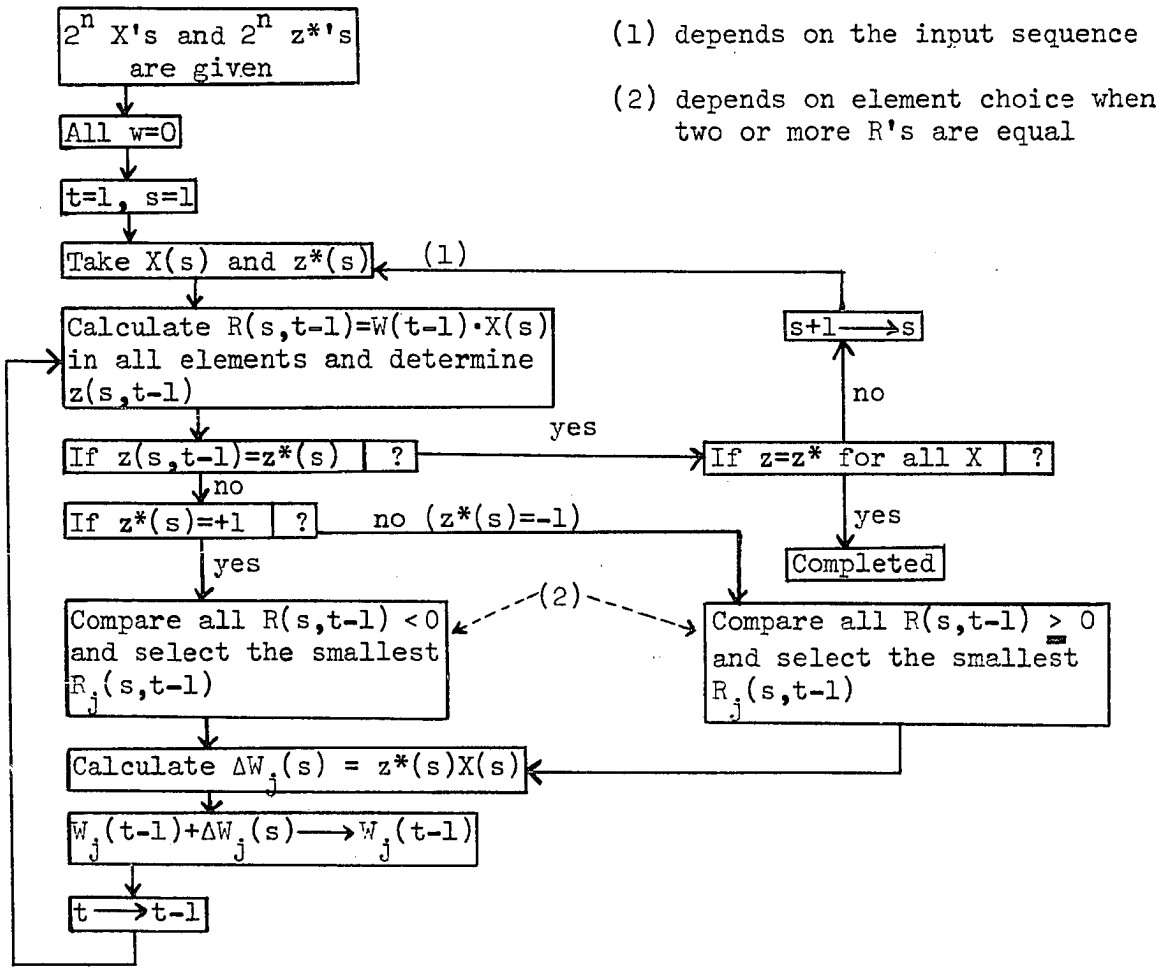


Diagram 2. Flow diagram of the simulation program for the universal learning network (d)

Four programs corresponding to the four networks in Figure 10 were written under the assumptions:

1. The truth table of a Boolean function, which is a set of 2^n input combinations and the corresponding desired outputs, is repeatedly given.

2. Each time it is given, the input combinations are presented in a different order so that the whole sequence is not cyclic. But when the truth table has been presented m times, each input combination has been given exactly m times. An example is: 0,1,2,3,4,5,6,7;0,2,4,5,1,3,5,7; 1,4,7,2,5,0,3,6;... etc. in a 3-input case. For some functions a different input sequence was used for comparison.

3. All weights are zero at the start.

The computer simulation confirmed that the learning process does converge in a reasonable number of steps for all the representative functions. The results are summarized in Table 5. The table shows the maximum, minimum and average total numbers of weight changes (N) and those of truth table presentations (M). The maximum value of weights assumed by elements is also shown. The average was taken over all 10 representative functions of 3-variables or over all 208 representative functions of 4 variables. When different input sequences or different choice of elements were tested, the smallest values of N and M were taken. The four-input OR network was not tested because it consists of 9 elements with a total of 40 parameters and it is considered too redundant.

The total number of weight changes may be considered to be a measure of learning efficiency and the number of parameters to be a measure of redundancy. Table 5 suggests:

1. When there is too much redundancy as in the OR network or in the symmetric network, learning efficiency is low.

2. When there is too little redundancy as in the Asymmetric network, learning efficiency is also low.

3. There is some optimum balance between the two factors as in the

Table 5. Results of computer simulation

	(a) OR		(b) Majority		(c) Symmetric		(d) Asymmetric	
	N	M	N	M	N	M	N	M
3-input networks	5 el[a] 16 pa[b]		4 el 12 pa		4 el 16 pa		2 el 9 pa	
Maximum	52	7	18	4	47	8	52	10
Minimum	5	2	9	1	8	2	7	1
Average	18	5	12	2	21	6	22	4
Max. w	4		4		4		4	
4-input networks	9 el 40 pa		5 el 20 pa		5 el 25 pa		3 el 17 pa	
Maximum	----		99	17	198	24	135	22
Minimum	----		13	2	25	4	11	2
Average	----		37	7	92	12	55	10
Max. w	----		7		6		10	

[a] el = The number of elements

[b] pa = The number of parameters

Majority network.

In the actual choice of a network, other factors such as economy and availability of inputs must be taken into account. A trade-off may have to be made somewhere. The symmetric network seems to have no advantage over the Majority network but it is of interest as an example of a network with multi-layers of exactly identical adaptive elements.

As for the Asymmetric network, its structure obtained as a result of learning is directly comparable with the structure listed in the table of R. C. Minnick (6), which was obtained by a linear programming technique.

The table shows the optimum structures of all 221 representative functions of up to four variables. As an example, consider f90(0,1,2,5,14,15) which is realizable with a 4-input network as shown in Figure 10(d). Two different structures were obtained by learning with a random and a deterministic choices of elements when two or more elements had the same reference level. The deterministic choice means a selection of an element with the lowest number among those elements, assuming that all elements in the network are numbered in some fashion. The two structures 1) and 2) obtained by learning are compared with the optimum structure (Opt.) taken from R. C. Minnick's table.

	$(w_{11},w_{12},w_{13},w_{14};T_1)$	$(w_{21},w_{22},w_{23},w_{24};T_2)$	$(w_{31},w_{32},w_{33},w_{34};T_3;w_1,w_2)$
Opt.	(-1, -1, -1, 0;+2)	(0, -1, 0, -1;+1)	(+2, +2, +1, +1;-2;+4,+2)
(1)	(+2, 0, -6, -2;-2)	(-2, -2, -2, 0;+6)	(+2, +2, +4, +2; 0;+4,+6)
(2)	(-2, +4, -2, 0;-2)	(-1, -5, -1, -3;+3)	(+5, +1, +3, +1;+3;+7,+7)

The first two sets of weights are those of two input elements. The third set of weights is those of the output element. Its last two weights are those associated with the outputs from the two input elements. For some functions the choice made in the equal reference level case has a significant effect also on both the total number of weight changes and the number of truth table presentations as well as on the final values of weights. Sometimes a purely random choice may lead to more rapid convergence or a more optimum representation. For example,

	Random Choice	Deterministic Choice
f81=f(0,1,2,5,6,12)	N= 39, M= 6	N=61, M=9
f159=f(0,1,2,7,11,12,15)	N=150, M=19	N=50, M=8

This suggests that there are many cases when two or more elements come to have the same reference level in the course of learning and the choice of an element in such cases can greatly affect the entire learning process.

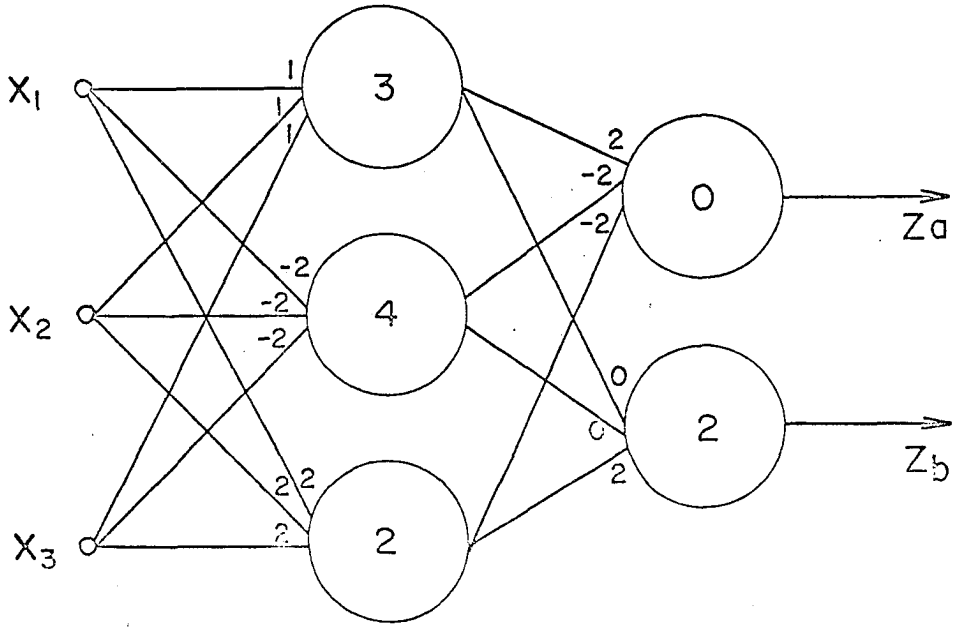
VI. LEARNING IN A MULTI-OUTPUT NETWORK

It may be expected that a learning procedure similar to that for a single-output network can be applied for a multi-output network. If a multi-output network has no adjustable parameters which are shared in common by more than one output element, the network can be divided into single-output networks in parallel. In this case, the network is a combination of essentially single-output networks and the learning procedure used in a single-output network is readily applicable. All multi-output learning networks so far used primarily for pattern recognition are this type of networks (1,3).

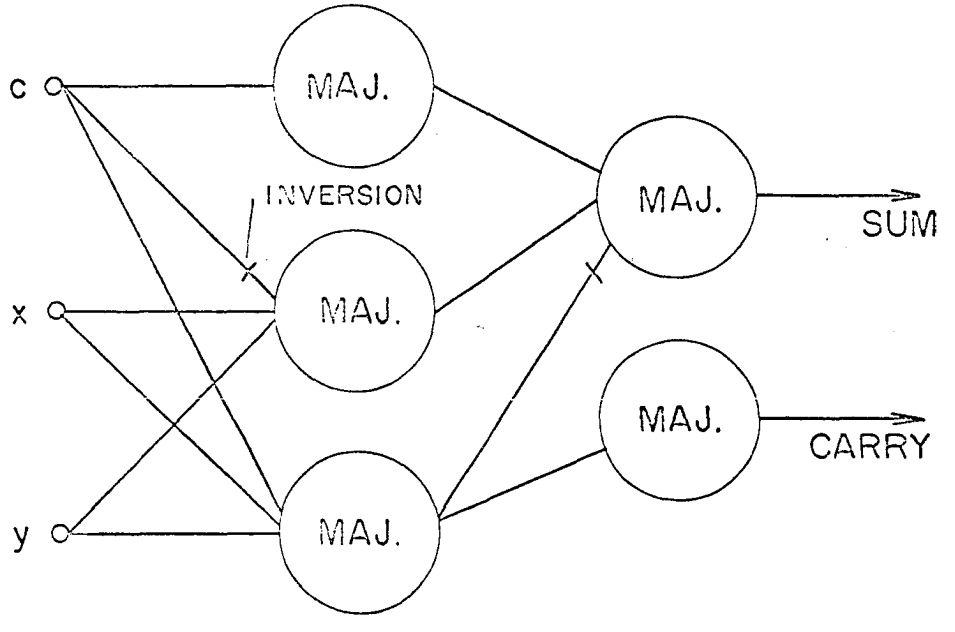
On the other hand, if a multi-output network has common adjustable parameters shared by more than one output element, the network may be called non-separable in the sense that it is not possible to divide the network into single-output networks in parallel. A new problem in such a non-separable network is the fact that a weight change made to alter only certain outputs can affect the other outputs. A weight change favorable to one output element may not be so for the other output elements. Actually it can be undesirable for them in some cases. The learning procedure must provide means to avoid this conflict of interest.

Though any complicated learning procedure is conceivable, a simple procedure is desirable. The following learning procedure can be applied for a 3-input 2-output network as shown in Figure 12(a). It is based on hierarchy among elements. All five elements in the network are assumed to be adaptive.

Figure 12. Two full adders



(a) 3-INPUT 2-OUTPUT LEARNING NETWORK



(b) CONVENTIONAL MAJORITY ELEMENT NETWORK

Learning Procedure (iv) for a multi-output network

1. When one or both outputs are in error, first an input element having the smallest absolute reference level is selected and its weights are changed so that the output of that element is reversed. As a result, if both outputs become correct, the network proceeds to take the next inputs.

2. If one or both outputs are still in error, then the weights of the output elements in error are changed so that both outputs become correct.

3. Other details are the same as in Learning Procedure (ii).

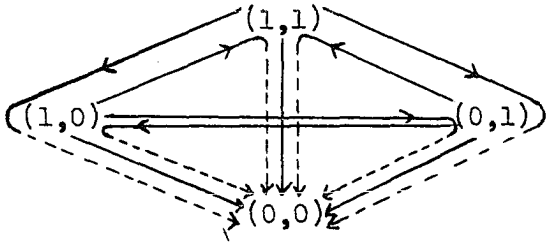
It is convenient to consider the learning procedure in terms of an error state transition chart. Such a chart is shown below, where

$(E_a,E_b)=(1,1)$ is the state where both z_a and z_b are in error.

$=(1,0)$ is the state where only z_a is in error.

$=(0,1)$ is the state where only z_b is in error.

$=(0,0)$ is the state where both z_a and z_b are correct.



A solid line shows a transition due to step 1 and a dotted line shows a transition due to step 2. The transitions causing no state changes, which are probable under step 1, are not shown. By the above learning procedure, all error states finally settle to the (0,0) state. The conflict of

interest is automatically avoided.

The convergence of this learning process in the 3-input 2-output network as shown in Figure 12(a) was tested by a computer simulation similar to those used in Chapter V. For comparison, consider a conventional full adder using Majoriy elements as shown in Figure 12(b) and a Boolean function combination f(1,2,4,7) for the sum function and f(3,5,6,7) for the carry function. The numbers in Figure 12(a) show the weight values obtained by learning for the function combination. The number in a circle (element) shows the threshold of that element. The full adder was realized after 101 weight changes and 12 truth table presentations. Other function combinations were also tested and were observed to be realizable after similar weight changes.

The learning process in this 2-output network is more lengthy, or requires more weight changes than in those single-output networks studied in the previous chapters. The mathematical proof of convergence in this process has not been established but it is conjectured that the process converges in a finite number of steps provided that no cyclic changes of weights occur. When the network becomes larger, the learning process will become more lengthy. It should be noted, however, that learning takes place automatically as long as a truth table is repeatedly given. It may be expected that a prescribed function combination will eventually be established by learning if a sufficient time is allowed for learning.

VII. CONCLUSIONS

In dealing with networks of threshold elements, the following conclusions are established.

1. A necessary condition for a Boolean function to be realizable with a single threshold element is that there be no isolated true term in the function.

2. A lower bound of the number of threshold elements necessary for a universal network was obtained in terms of the number of inputs.

3. Various types of universal networks consisting of threshold elements were compared in terms of the number of parameters.

4. The computer simulation of learning process in a network gives a method of determining appropriate weight values to realize a function.

The analysis of the learning process has led to the following conclusions.

5. The length of a weight vector changes approximately proportionally to $\sqrt{t}$, where t is the number of weight changes.

6. A nonlinear threshold element was proposed as a new element. Though the element may have a complicated structure, it has merits of tremendous versatility and easy training.

7. Simple learning procedures suitable for multi-element networks were established. Necessary conditions for convergence were pointed out. The convergence property was confirmed by a computer simulation for networks with small number of inputs.

8. A hierarchy consideration is useful in a network with more than one layer of adaptive elements.

9. It was demonstrated that, though redundancy is necessary for learning, too much redundancy tends to lower learning efficiency.

A 5-input threshold element, 4-input universal networks and a 3-input 2-output network were tested for the convergence of their learning process. All these networks are considered physically realizable with a reasonable effort. The results obtained by the computer simulation of these networks will be useful in checking experimental data on those physical networks.

In this thesis, universal networks were mainly considered. It was assumed that a complete truth table was given during learning. These are very severe requirements for a learning network. It is clear that, if the requirement of universality is dropped, the network will become simpler or the number of inputs may easily be increased. If a separable multi-output network is considered, the number of outputs may also easily be increased. Such a network can serve as a pattern recognition system and the same learning procedure as developed in this thesis may be used. When only an incomplete truth table is to be given, that is, particular input combinations never occur, the network will exhibit a generalizing property, because the network can respond in some way to the input combinations which were not given during learning period. It is also possible to investigate ternary logic networks by treating such absent input combinations as don't-care conditions.

The fact that the initial condition cannot be arbitrary in a multi-element learning network suggests that a decay factor in each weight may be effective in permitting arbitrary initial conditions and also possibly cyclic input sequences, because the weights will never change cyclically for a cyclic input sequence due to the decay factor.

Though, in principle, the learning procedures are applicable to much larger networks than the networks considered in this thesis, the learning process will become very complicated and lengthy in such large networks. It may be necessary to break such a network into sub-networks and to supply some information concerning the intermediate states between the sub-networks.

The learning networks having a small number of inputs will find great practical applications when they may be connected to conventional logical networks. There may be many situations where a large number of input signals could be processed first by a prewired logical network and reduced to a small number of signals. Those signals could then be fed into a universal learning network which could be trained to respond to a changing demand.

VIII. LITERATURE CITED

1. Rosenblatt, F. Principles of neurodynamics. Washington, D.C., Spartan Books, Inc. 1961.

2. Learning by threshold elements in cascade. Newport Beach, California, Aeroneutronic, Ford Motor Co. 1961.

3. Ridgway, W. C., III. An adaptive logic system with generalizing properties. Stanford University Electronics Laboratory Technical Report No. 1556-1. 1962.

4. Muroga, S., Toda, I. and Takasu, S. Theory of majority decision functions. Franklin Institute Journal 271: 376-418. 1961.

5. Harrison, M. A. The number of equivalence classes of Boolean functions under groups containing negation. Institute of Electrical and Electronics Engineers Transactions on Electronic Computers EC-12: 559-561. 1963.

6. Minnick, R. C. Linear input logic. Institute of Radio Engineers Transactions on Electronic Computers EC-10: 6-16. 1961.

7. Winder, R. O. Bounds on threshold gate realizability. Institute of Electrical and Electronics Engineers Transactions on Electronic Computers EC-12: 561-564. 1963.

8. Takahasi, H. Computing machines. In Iwanami modern applied mathematics series. (In Japanese) Vol. B-14-a. pp. 57-58. Tokyo, Japan, Iwanami Books, Inc. 1958.

IX. ACKNOWLEDGEMENTS

The author wishes to express his sincere thanks to his major professor, Dr. Robert M. Stewart, Jr. for many enlightning suggestions. He also wishes to thank the staff members of the Cyclone Computer Laboratory for keeping the Cyclone Digital Computer in an excellent operating condition while it was in his use. Thanks are also due to Mr. Karl H. G. Sera for helping the author prepare the manuscript, to Mrs. Natalie T. Skola for typewriting the final manuscript, and to Mr. William D. Thomas for doing the drawings.