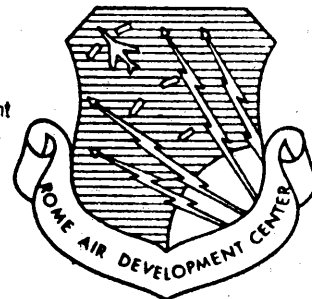


File Copy - Do not remove

0-8127

RADC-TR-65-257
Final Report

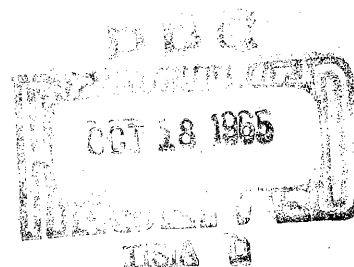
Nils J. Nilsson
Computer Science Department
Stanford University
Stanford, CA. 94305-4110



THEORETICAL AND EXPERIMENTAL INVESTIGATIONS
IN TRAINABLE PATTERN-CLASSIFYING SYSTEMS

Nils J. Nilsson
(Stanford Research Institute)

TECHNICAL REPORT NO. RADC-TR-65-257
September 1965



Information Processing Branch
Rome Air Development Center
Research and Technology Division
Air Force Systems Command
Griffiss Air Force Base, New York

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

**THEORETICAL AND EXPERIMENTAL INVESTIGATIONS
IN TRAINABLE PATTERN-CLASSIFYING SYSTEMS**


**Nils J. Nilsson
(Stanford Research Institute)**


FOREWORD

The research described in this report grew out of an earlier RADC project under Contract AF30(602)-2943 entitled "Linear Separability of Signal Space as a Basis for Adaptive Mechanisms." During this earlier project, the properties of several trainable pattern classifiers were studied. These classifiers included TLU's, committee machines, linear machines and piecewise linear (PWL) machines. The results of these studies were reported in Technical Documentary Report No. RADC-TDR-64-145 (May 1964).


This report was prepared by Stanford Research Institute, Menlo Park, California, under Contract AF30(602)-3448, Project 5581, Task 558104. RADC Project Engineer was Mr. Charles Constantino (EMIID).

Several individuals contributed to the research performed during this project. C.A. Rosen, D.J. Hall, and G. Ball did much of the work on mode-seeking training. Consultant T. Cover was responsible for research on classification capacity and the nearest neighbor decision rules. D.J. Kaylor continued work on committee machines and conducted research on PWL machine error-correction training methods. The material in Sec. III was adapted from a paper by R.O. Duda and H. Fossum describing their research performed under this project.

Approved: 
FRANK J. TOMAINI
Chief, Info Processing Branch

Approved: 
ROBERT J. QUINN, JR.
Colonel, USAF
Chief, Intel and Info Processing Div.

FOR THE COMMANDER:


IRVING J. GABELMAN
Chief, Advanced Studies Group

ABSTRACT

Motivation is given for the use of various trainable pattern-classifying structures called linear and piecewise linear (PWL) machines. The results of various experiments in training these machines are presented. Two different types of training methods were investigated: mode-seeking and error-correction methods. These methods are illustrated by experiments using two-dimensional patterns so that the results of training can be easily visualized. More thorough experiments with 10-dimensional and 100-dimensional patterns are also described.

It is concluded that certain of these training methods can be expected to give good performance in complex pattern classifying tasks involving multi-modal pattern distributions. The report concludes with a list of recommendations for future research, and contains an Appendix presenting a new theorem on training a linear machine.

CONTENTS

ABSTRACT.	ii
LIST OF ILLUSTRATIONS	v
LIST OF TABLES.	viii
ACKNOWLEDGMENTS	ix
 I TRAINABLE PATTERN CLASSIFIERS.	 1
A. Description of the Problem.	1
1. Pattern Points and Hypersurfaces	1
2. Adjustable Hypersurfaces	2
B. Hypersurface Selection Methods Motivated by Decision Theory	2
1. "Optimum" Classifiers.	2
2. The Fix and Hodges Classifier.	4
3. The Nearest-Neighbor Decision Rule	5
4. Nearest-Mode Classifiers	8
C. Structure of Nearest-Mode and Related Classifiers .	10
1. The Dot-Product Unit	10
2. Piecewise Linear Machines.	13
D. Other Classifying Machines.	15
1. Dichotomizers.	15
2. Parallel Organization of Dichotomizers	21
3. Φ Processors	26
E. The Problem of Training Pattern-Classifying Machines.	27

ILLUSTRATIONS

Fig. 1	A Dot-Product Unit.	12
Fig. 2	Boundary Surfaces for a Nearest-Mode Classifier . .	14
Fig. 3	Boundary Surfaces for a PWL Machine	16
Fig. 4	An R-Category PWL Machine	17
Fig. 5	An R-Category Linear Machine.	18
Fig. 6	A Threshold Logic Unit (TLU).	20
Fig. 7	A Committee Machine	22
Fig. 8	An R-Category Classifier Using Parallel Dichotomizers.	23
Fig. 9	Patterns for Mode-Seeking Experiments	35
Fig. 10	Nine Weight Vectors Found Using Stark's Method. . .	38
Fig. 11	Result of First Cycle of ISODATA Process.	42
Fig. 12	Result of Second Cycle of ISODATA Process	44
Fig. 13	Result of Third Cycle of ISODATA Process.	45
Fig. 14	Final Result after Four Cycles of ISODATA Process .	46
Fig. 15	Pattern Categories for Rosen-Hall Method Experiment.	50
Fig. 16	Result of First Cycle of Rosen-Hall Method.	52
Fig. 17	Result of Second Cycle of Rosen-Hall Method	53
Fig. 18	Final Positions of Weight Vectors after Third Cycle of Rosen-Hall Method.	54
Fig. 19	PWL Machine Used in Error-Correction Training Experiment with Two-Dimensional Patterns.	59
Fig. 20	Plot of Corrections vs Cycle Number for Error-Correction Training of a PWL Machine.	61

Fig. 21	Decision Surfaces after 19th Cycle.	62
Fig. 22	Performance of a Minimum-Distance Classifier on Unimodal Gaussian Data.	71
Fig. 23	Waveform Representation of Prototype Pattern Vectors	72
Fig. 24	Performance of a Linear Machine Trained on Unimodal Spherical Data.	79
Fig. 25	Experimental Results (Percentage Error on Testing Patterns) for Gaussian Data	80
Fig. 26	Effect of Sample Size on Performance.	84
Fig. 27	Performance of Committee Machines Trained on Multi- Modal, Quantized Gaussian Data.	86

II	TRAINING METHODS	30
A.	Linear Machines	30
B.	Piecewise Linear Machines	32
1.	Mode-Seeking Training.	32
2.	Error-Correction Training.	55
C.	Transferring Weight Vectors Among Banks During Training.	64
III	ERROR-CORRECTION TRAINING EXPERIMENTS.	66
A.	Purpose of the Experiments.	66
B.	The Ten-Dimensional Gaussian Patterns	67
C.	The Error-Correction Rule Used with the Linear and PWL Machines.	70
D.	Experiments Using the Ten-Dimensional Gaussian Patterns.	76
E.	Experiments with 100-Dimensional, Quantized Gaussian Patterns	82
F.	Experiments with Committee Machines	83
IV	SUMMARY AND CONCLUSIONS.	88
	APPENDIX.	91
	REFERENCES.	101
	TECHNICAL DOCUMENTARY REPORTS PREPARED DURING THE PROGRAM . .	103

TABLES

Table 1	Subclass Assignments for Multimodal Data	69
---------	--	----

I TRAINABLE PATTERN CLASSIFIERS

A. DESCRIPTION OF THE PROBLEM

1. Pattern Points and Hypersurfaces

Traditional statements of the pattern-classification problem have relied on a geometric formulation that has proven to be quite useful.^{1*} We begin this report with a brief recapitulation of this formulation, since it will be used throughout.

We assume that the object to be classified is first subjected to certain measurements. (Some examples of objects are: visual images, human voice, radar echos, meteorological conditions, and diseases.) The result of these measurements is a set of measurement values. Let us assume that there are d measurements made and that the values of these measurements are denoted by the quantities x_1, x_2, \dots, x_d . We shall call such a set of measurement values a pattern. It is convenient to represent a pattern by a point, \vec{X} , in a d -dimensional space. The coordinates of the point are the values x_1, x_2, \dots, x_d . Alternatively, it will sometimes be convenient to represent the pattern by the vector \vec{X} with components x_1, x_2, \dots, x_d .

Now we desire to operate on these measurement values to determine the category of the pattern \vec{X} (and thus of the object under test). Suppose the object can belong to one of R categories and that we want the result of our operation on \vec{X} to be a (non-random) decision about the category of the object. The rule by which such decisions are made can be described geometrically by referring to hypersurfaces that divide the space of patterns into R distinct regions, one region per

* References are listed at the end of this report.

category. (These regions could be disjoint). The problem of designing a device (or of writing a computer program) to classify patterns can thus be phrased geometrically as the problem of placement of hypersurfaces in a pattern space.

2. Adjustable Hypersurfaces

There are many circumstances in which it is unreasonable to attempt to specify the locations and types of the hypersurfaces without first examining some typical patterns whose categories are known. Such a set of typical patterns we shall call a training set. Any method of selection and/or placement of hypersurfaces employing a training set we shall call a training method. Often it is convenient to implement a pattern classifier whose hypersurfaces are adjustable. Such a classifier we shall call a trainable pattern classifier. The central problem considered during this project has been the following: Given a trainable pattern classifier and a training set of patterns, how should the hypersurfaces be selected so that the classifier performs adequately on the training-set patterns? Inasmuch as most of the research effort of this project has attempted to answer this question, the rest of this report will be mainly concerned with tracing the development of some possible answers and presenting some experimental evidence of their utility.

B. HYPERSURFACE SELECTION METHODS MOTIVATED BY DECISION THEORY

1. "Optimum" Classifiers

It is sometimes assumed that the patterns to be categorized are random vectors. If so, for each pattern category there will exist a probability function (or probability density function) describing

the way in which the patterns belonging to that category are distributed throughout the pattern space. Suppose we denote these functions by $p(\vec{X}|i)$ for $i=1, \dots, R$. $p(\vec{X}|i)$ is the probability (or probability density) that a pattern selected from category i will be \vec{X} . Let the a priori probability of a pattern belonging to category i be given by $p(i)$ for $i = 1, \dots, R$. A result of statistical decision theory² is that to minimize the probability of making erroneous classifications, the optimum classifier should operate as follows: Suppose the pattern to be classified is \vec{X} . The expressions

$$g_i(\vec{X}) = p(\vec{X}|i) p(i) \quad (1)$$

are evaluated for $i=1, \dots, R$, and \vec{X} is placed in that category, i_0 , corresponding to the largest of these expressions. The hypersurfaces in the pattern space are thus given by expressions of the form

$$p(\vec{X}|i)p(i) = p(\vec{X}|j)p(j) \quad (2)$$

$$i, j = 1, \dots, R, \quad i \neq j.$$

Unfortunately, Eq. (1) cannot be used by a pattern classifier if the $p(\vec{X}|i)$ and $p(i)$, for $i=1, \dots, R$, are not known. We are interested here in precisely those situations in which these expressions are not known. In such situations one possibility is to use the patterns in the training set to estimate $p(\vec{X}|i)$ and $p(i)$, $i=1, \dots, R$. Such a procedure is reasonable and indeed motivates some of the techniques and experiments that will be presented in this report. We shall begin our

discussion of trainable pattern classifiers by presenting one method of estimating $p(\vec{X}|i)p(i)$ for $i=1,\dots,R$.

2. The Fix and Hodges Classifier

Suppose we have a set of training patterns whose categories are known. The Fix and Hodges³ classifier uses these training patterns to classify an unknown pattern \vec{X} as follows: Select an integer, k , and collect the k closest training patterns to \vec{X} ("closeness" can be measured by Euclidean distance). Suppose that, of these k closest patterns, n_1 patterns belong to category 1, n_2 to category 2, ..., and n_R to category R , ($n_1 + n_2 + \dots + n_R = k$). Then set

$$\begin{aligned} g_1(\vec{X}) &= n_1 \\ g_2(\vec{X}) &= n_2 \\ &\vdots \\ g_R(\vec{X}) &= n_R \end{aligned} \tag{3}$$

and decide in favor of that category number, i_0 , corresponding to the largest of these numbers.

The Fix and Hodges method clearly is an attempt to estimate a set of numbers proportional to $p(\vec{X}|i)p(i)$ for $i=1,\dots,R$ around the point \vec{X} . If such a set of numbers is well approximated by the set n_1, n_2, \dots, n_R , then the specification of the $g_i(\vec{X})$ by Eq. (3) will lead to nearly optimum (minimum error probability) classifications.

Selection of the integer, k , is quite important in the application of the Fix and Hodges procedure. If k is too small, the $g_i(\vec{X})$ will

vary rapidly with \vec{X} , and the consequent decision surfaces will be highly sensitive to the spatial locations of the training patterns.

If k is too large (in relation to the number of patterns in the training subsets), the $g_i(\vec{X})$ will not be sensitive enough to the actual variations of the probability distributions with \vec{X} . If the training subsets are large, it has been shown³ that the Fix and Hodges classifier leads to the same classifications as would be made if the (unknown) probability distributions were known and used. In general, the value of k should increase without limit with increasing N , if N is the total number of patterns in the training subsets. The value of k/N , however, should decrease toward zero with increasing N .

Even when k is extremely small, however, the Fix and Hodges method still leads to classifications comparing favorably with those made by the optimum (minimum probability of error) classifier in the limit of very large training sets. In the next part we shall show that this comparison is still quite favorable even when $k = 1$.

3. The Nearest-Neighbor Decision Rule^{*}

When $k = 1$, the Fix and Hodges classifier places an unknown pattern, \vec{X} , in the same category as that of the closest pattern in the training set. Such a rule results in what is called a nearest-neighbor classifier. The nearest-neighbor classifier can be shown to be reasonably

* The results of this section are due to Prof. T. Cover (consultant to the Institute) and Mr. P. Hart of Stanford University.

effective compared with the optimum classifier. We shall outline the steps of the argument in the case in which there are just two categories of patterns ($R = 2$).

Suppose the probability density functions $p(\vec{X}|1)$ and $p(\vec{X}|2)$ are continuous functions of \vec{X} . Then if $p(1)$ and $p(2)$ are the a priori probabilities of Categories 1 and 2, respectively, we have by Bayes rule

$$p(i|\vec{X}) = \frac{p(\vec{X}|i)p(i)}{p(\vec{X})} \quad (4)$$

for $i = 1$ and 2 , where $p(i|\vec{X})$ is the probability that pattern \vec{X} belongs to Category i . The classifier that achieves the minimum probability of error determines the larger of $p(1|\vec{X})$ and $p(2|\vec{X})$. Its probability of error, $PE(\vec{X})$, for any \vec{X} is therefore given by

$$PE(\vec{X}) = \min[p(1|\vec{X}), p(2|\vec{X})] \quad (5)$$

The nearest neighbor classifier determines the category of the nearest neighbor. For continuous $p(\vec{X}|i)$, $i=1$ and 2 , and in the limit of an infinitely large training set

$$\begin{aligned} &[\text{Probability that nearest neighbor to } \vec{X} \text{ belongs} \\ &\text{to category } i] = p(i|\vec{X}) \end{aligned} \quad (6)$$

for $i = 1$ and 2 . The probability $PE^*(\vec{X})$ that the nearest neighbor rule leads to an erroneous categorization of \vec{X} is given by

$$\begin{aligned}
PE^*(\vec{X}) &= \left[\begin{array}{l} \text{Probability that nearest neighbor} \\ \text{to } \vec{X} \text{ belongs to category 1} \end{array} \right] p(2|\vec{X}) \\
&+ \left[\begin{array}{l} \text{Probability that nearest neighbor} \\ \text{to } \vec{X} \text{ belongs to category 2} \end{array} \right] p(1|\vec{X}) \\
&= 2p(1|\vec{X})p(2|\vec{X}) \quad . \quad (7)
\end{aligned}$$

Since $p(1|\vec{X}) = 1 - p(2|\vec{X})$, we have

$$PE^*(\vec{X}) = 2p(1|\vec{X})[1 - p(1|\vec{X})] \quad (8)$$

By considering the two possibilities $p(1|\vec{X}) > p(2|\vec{X})$ and $p(1|\vec{X}) < p(2|\vec{X})$ in Eq. (5), the reader can easily verify that we may write

$$PE^*(\vec{X}) = 2PE(\vec{X})[1 - PE(\vec{X})] \quad . \quad (9)$$

$PE(\vec{X})$, of course, depends on \vec{X} . Since we assume \vec{X} is a random variable, $PE(\vec{X})$ can be averaged to give a global probability of error PE . We then have for the optimum classifier

$$PE = E[PE(\vec{X})] \quad (10)$$

and for the nearest neighbor classifier

$$PE^* = E[PE^*(\vec{X})] \quad , \quad (11)$$

where $E[\]$ denotes the expectation operator

Using Eq. (9) we have

$$PE^* = 2PE - 2E[PE^2(\vec{X})] \quad (12)$$

Since the variance of $PE(\vec{X})$ can be written

$$\text{var } PE(\vec{X}) = E[PE^2(\vec{X})] - E^2[PE(\vec{X})] \quad (13)$$

we have

$$PE^* = 2PE - 2 \text{ var } PE(\vec{X}) - 2(PE)^2 \quad (14)$$

or

$$PE^* \leq 2 PE(1-PE) \quad (15)$$

with equality holding only when $\text{var } PE(\vec{X}) = 0$.

The inequality expressed by Eq. (15) states that the asymptotic probability of error of the nearest neighbor classifier is never higher than twice the probability of error of the optimum classifier. The fact that one often need not tolerate an excessively larger probability of error using the nearest neighbor classifier makes it a candidate for use in those situations in which the optimum classifier cannot be designed for lack of knowledge of $p(\vec{X}|i)p(i)$.

4. Nearest-Mode Classifiers

The Fix and Hodges classifier (including the nearest neighbor classifier) appears superficially to be a reasonable answer to the problem of pattern classification. These classifiers are trainable be-

cause the type and location of the separating hypersurfaces depend on the patterns in the training set.

The Fix and Hodges classifier (even when $k=1$) suffers one important drawback, however. To classify any pattern, \vec{X} , the distance between \vec{X} and each of the patterns in the training set must be computed. If these computations are to be performed rapidly, each of the training patterns must be stored in some rapid-access memory. Because the method works best when the number of training patterns is large, the rapid-access storage requirements are often excessive. This disadvantage motivates a search for other methods that preserve some of the features of the Fix and Hodges classifier without requiring the individual storage of every training pattern in a rapid-access memory.

Rather than storing a large number of training patterns, we might store only a few "typical" training patterns. Each typical pattern selected for storage might actually represent many training patterns that cluster around it in the pattern space. Thus each typical pattern for a given category might be thought of as a "mode" of the probability density function for that category. A mode of a probability density function is the location of a local maximim of that function. We shall often attempt to estimate this location by a center of gravity of a finite cluster of training patterns. Thus we shall often speak loosely of a center of gravity as a mode.

The decision rule to be described assumes the existence of a method to find good estimates for these modes, given the training subsets. Suppose the modes for the various categories, as established by a training procedure, are given by the points $\vec{P}_i^{(j)}$ for $i = 1, \dots, R$ and

$j = 1, \dots, L_1$. That is, there are L_1 modes belonging to category 1, L_2 belonging to category 2, etc. Then, given these modes, one reasonable way to classify some arbitrary pattern \vec{X} is to measure its distance to each of the modes and place it in that category having the nearest mode. A classifier operating according to this principle will be called a nearest-mode classifier. Since the number of modes is usually much smaller than the total number of training patterns from which they were established, the rapid-access storage requirements for this method are much less demanding than those for the Fix and Hodges classifier. The concept of distance still plays an important role in a way which preserves some of the features of the Fix and Hodges method. It seems reasonable to assume that the k closest training patterns to a given pattern, \vec{X} , will often include a predominant number of patterns from the cluster surrounding the closest mode. Thus the nearest mode classifier will often make decisions identical to those made by the Fix and Hodges method.

C. STRUCTURE OF NEAREST-MODE AND RELATED CLASSIFIERS

1. The Dot-Product Unit

Suppose the modes are given by the points $\vec{P}_i^{(j)}$ for $i=1, \dots, R$ and $j=1, \dots, L_i$. The nearest mode classifier categorizes an unknown pattern \vec{X} by calculating the distances $|\vec{X} - \vec{P}_i^{(j)}|$ and finding the smallest of these.

Comparing the squared distances $|\vec{X} - \vec{P}_i^{(j)}|^2$ leads to the same classification. We may write these squared distances as follows

$$|\vec{X} - \vec{P}_i^{(j)}|^2 = \vec{X} \cdot \vec{X} - 2\vec{X} \cdot \vec{P}_i^{(j)} + \vec{P}_i^{(j)} \cdot \vec{P}_i^{(j)} \quad (16)$$

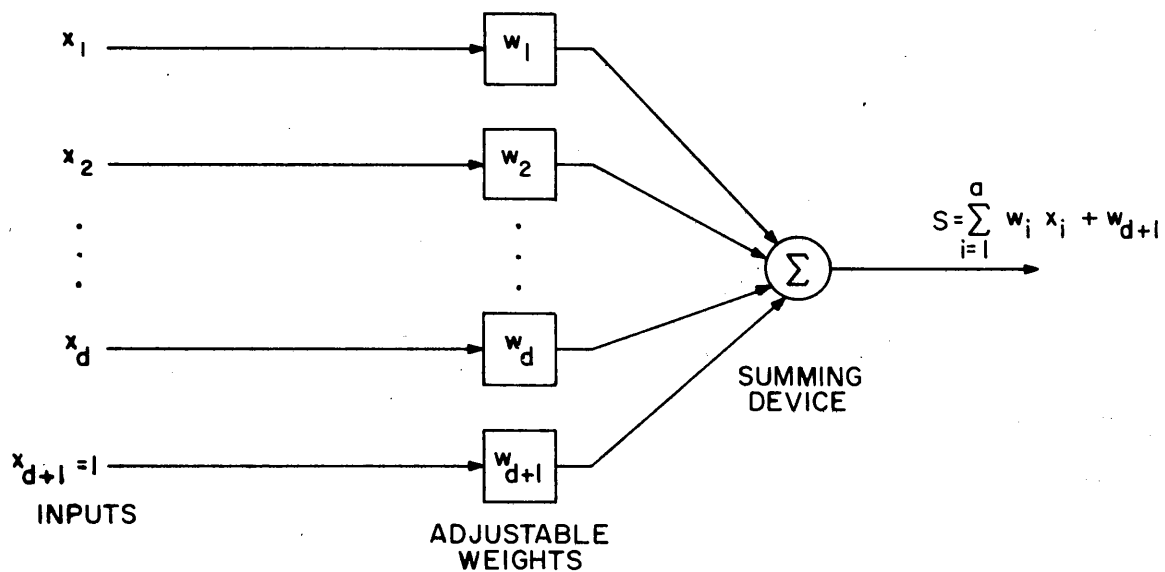
The term $\vec{X} \cdot \vec{X}$ is the same for all i and j and thus does not influence the comparison. We may multiply the remaining terms by $-\frac{1}{2}$ to obtain the expression

$$g_i^{(j)}(\vec{X}) = \vec{X} \cdot \vec{P}_i^{(j)} - \frac{1}{2} \vec{P}_i^{(j)} \cdot \vec{P}_i^{(j)} \quad (17)$$

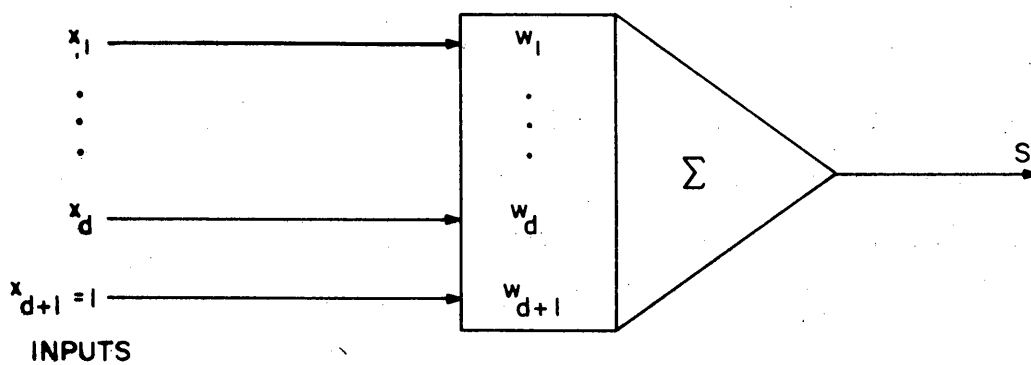
The smallest distance now corresponds to the largest $g_i^{(j)}(\vec{X})$. Thus the nearest mode classifier calculates all the $g_i^{(j)}(\vec{X})$ for $i=1, \dots, R$ and $j=1, \dots, L_i$ and places \vec{X} in that category whose index i is associated with the largest $g_i^{(j)}(\vec{X})$.

Examination of Eq. (17) reveals that each $g_i^{(j)}(\vec{X})$ consists of a constant plus a dot product between the pattern vector and the mode vector. It will be convenient to imagine that this expression is computed by a device we shall call a dot-product unit (DPU).^{*} The DPU is illustrated in Fig. 1. For patterns with d components, the DPU has $(d+1)$ adjustable "weights" $w_1, w_2, \dots, w_d, w_{d+1}$ and computes a weighted sum, $S = \sum_{i=1}^d w_i x_i + w_{d+1}$, of the pattern components. The first d weights, represented by a weight vector \vec{W} , correspond to the d pattern components, and the extra $(d+1)$ th weight allows a "bias" term to be added to the sum. It is often convenient to imagine that this $(d+1)$ th weight multiplies a fictitious $(d+1)$ th pattern component which is set permanently equal to some convenient value such as +1. In a nearest-mode classifier, each $g_i^{(j)}(\vec{X})$, as given by Eq. (17), is

*The DPU may be implemented by special electronic circuitry or its computation performed by a general-purpose digital computer.



(a)
ELEMENTS OF A DOT-PRODUCT UNIT



(b)
SCHEMATIC FOR A DOT-PRODUCT UNIT

FIG. 1 A DOT-PRODUCT UNIT

computed by a DPU. Each weight vector is set equal to the corresponding $\vec{P}_i(j)$ and the corresponding $(d+1)$ th weights are set equal to $-\frac{1}{2} \vec{P}_i(j) \cdot \vec{P}_i(j)$.

The hypersurfaces of a nearest-mode classifier are given by equations of the form

$$\vec{X} \cdot \left[\vec{P}_i(j) - \vec{P}_\ell(k) \right] - \frac{1}{2} \left[\vec{P}_i(j) \cdot \vec{P}_i(j) - \vec{P}_\ell(k) \cdot \vec{P}_\ell(k) \right] = 0 \quad (18)$$

Equation (18) is an equation of a hyperplane. This hyperplane is the perpendicular bisector of the line segment joining $\vec{P}_i(j)$ and $\vec{P}_\ell(k)$. Some example surfaces for two-dimensional patterns are illustrated in Fig. 2.

When there is only one mode per category, the nearest mode classifier employs just R DPU's (for R categories). Each DPU computes a dot product of the input pattern with a stored reference pattern representative of the category. Such a classifier is often said to employ "standard template matching" or "matched filter" techniques. The generalization to more than one DPU per category enhances the utility and performance of the classifier in those situations for which there is more than one representative pattern per category. Such "multimodal" situations are prevalent in pattern-recognition tasks.

2. Piecewise Linear Machines

We have said that the nearest mode classifier employs boundary hyperplanes that perpendicularly bisect the line segments joining the modes. The hyperplanes are perpendicular to the line segments because the weight vector of each DPU is made equal to a mode

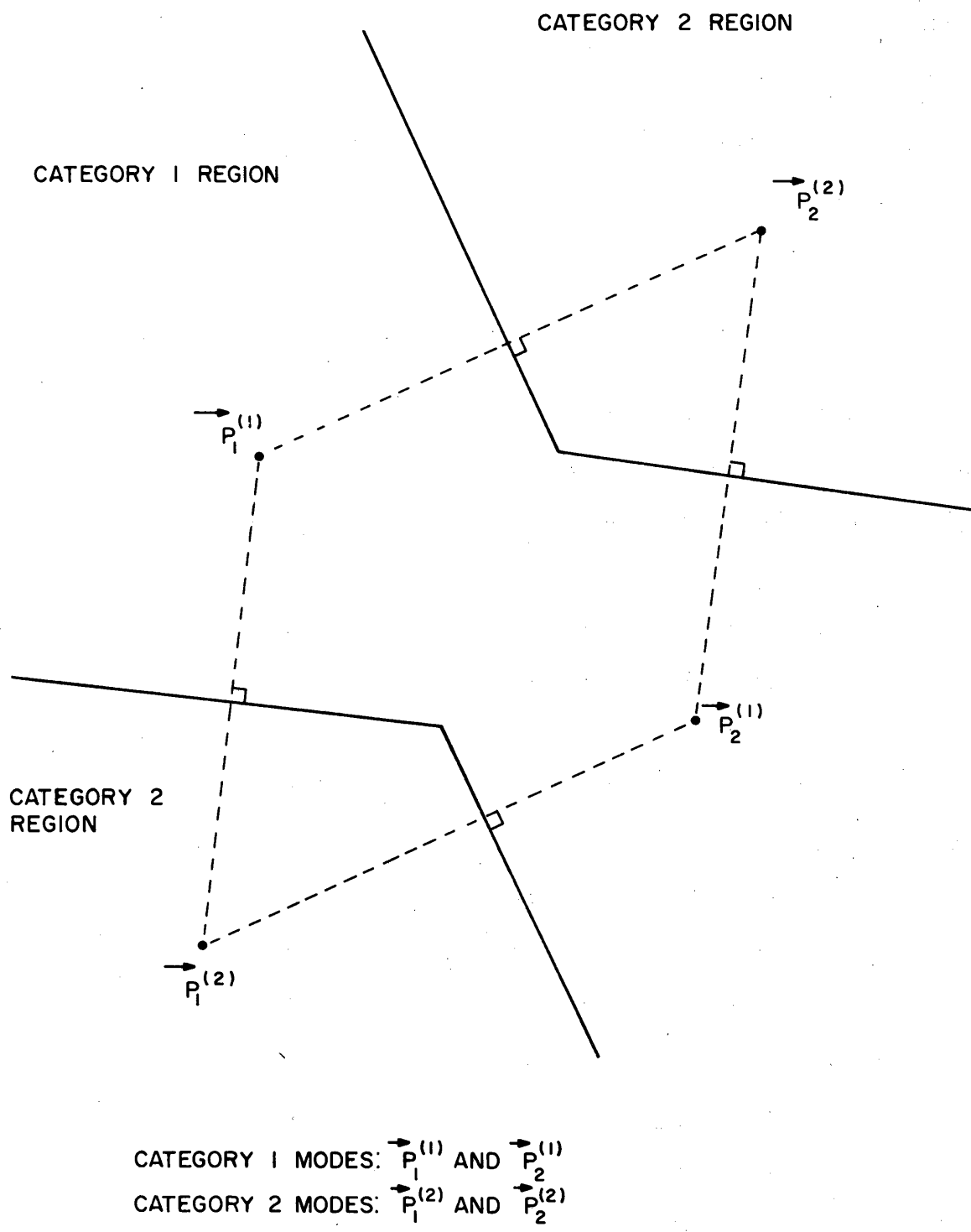


FIG. 2 BOUNDARY SURFACES FOR A NEAREST-MODE CLASSIFIER

vector; they bisect the line segments because each $(d+1)$ th weight is made equal to $-\frac{1}{2}$ the squared length of a mode vector. If the $(d+1)$ th weights only are allowed to change, the hyperplanes will remain perpendicular to line segments joining modes but will not necessarily bisect them. (In fact, they might not cross the line segments themselves, but may cross their extensions instead.) The resulting machine is no longer a nearest mode classifier; we have called it a piecewise linear (PWL) machine. A typical set of boundary surfaces for a PWL machine is illustrated in Fig. 3. The machine is called piecewise linear because the boundary surfaces separating any two categories consist of segments of hyperplanes.

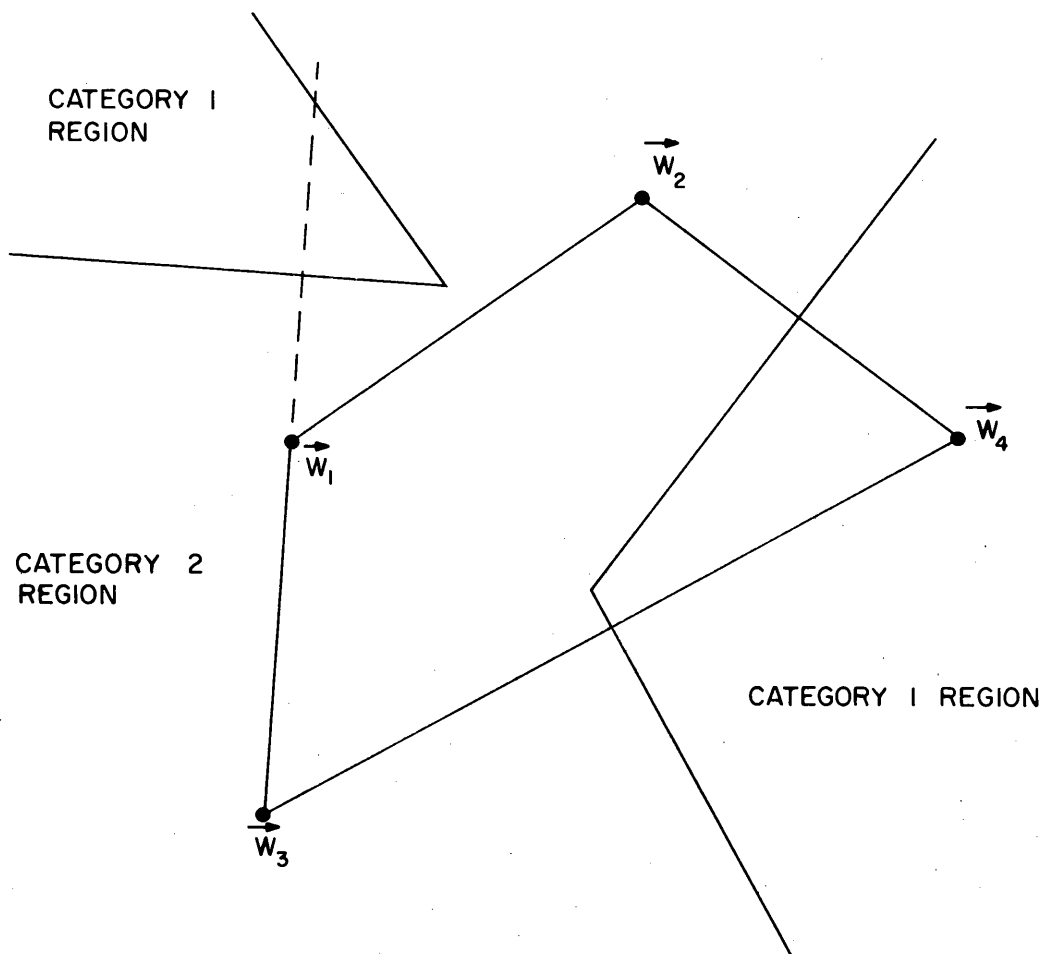
The implementation of a PWL machine is illustrated in Fig. 4. A PWL machine consists of R banks of DPU's--one bank for each category. A maximum selector indicates which bank contains the DPU with the largest output. If the i_0 -th bank contains the largest dot product, the input pattern is assigned to category i_0 .

If each bank contains only one DPU we have a special case of a PWL machine which we call a linear machine. A linear machine implements linear (i.e. hyperplane) boundary hypersurfaces separating two categories. Each of the R regions thus formed in the pattern space is an intersection of halfspaces; therefore these regions are each convex. The linear machine is illustrated in Fig. 5.

D. OTHER CLASSIFYING MACHINES

1. Dichotomizers

So far we have discussed the PWL machine and some special cases of it: the nearest mode classifier and the linear machine.



CATEGORY 1 WEIGHT VECTORS: \vec{w}_1, \vec{w}_4
 CATEGORY 2 WEIGHT VECTORS: \vec{w}_2, \vec{w}_3

FIG. 3 BOUNDARY SURFACES FOR A PWL MACHINE

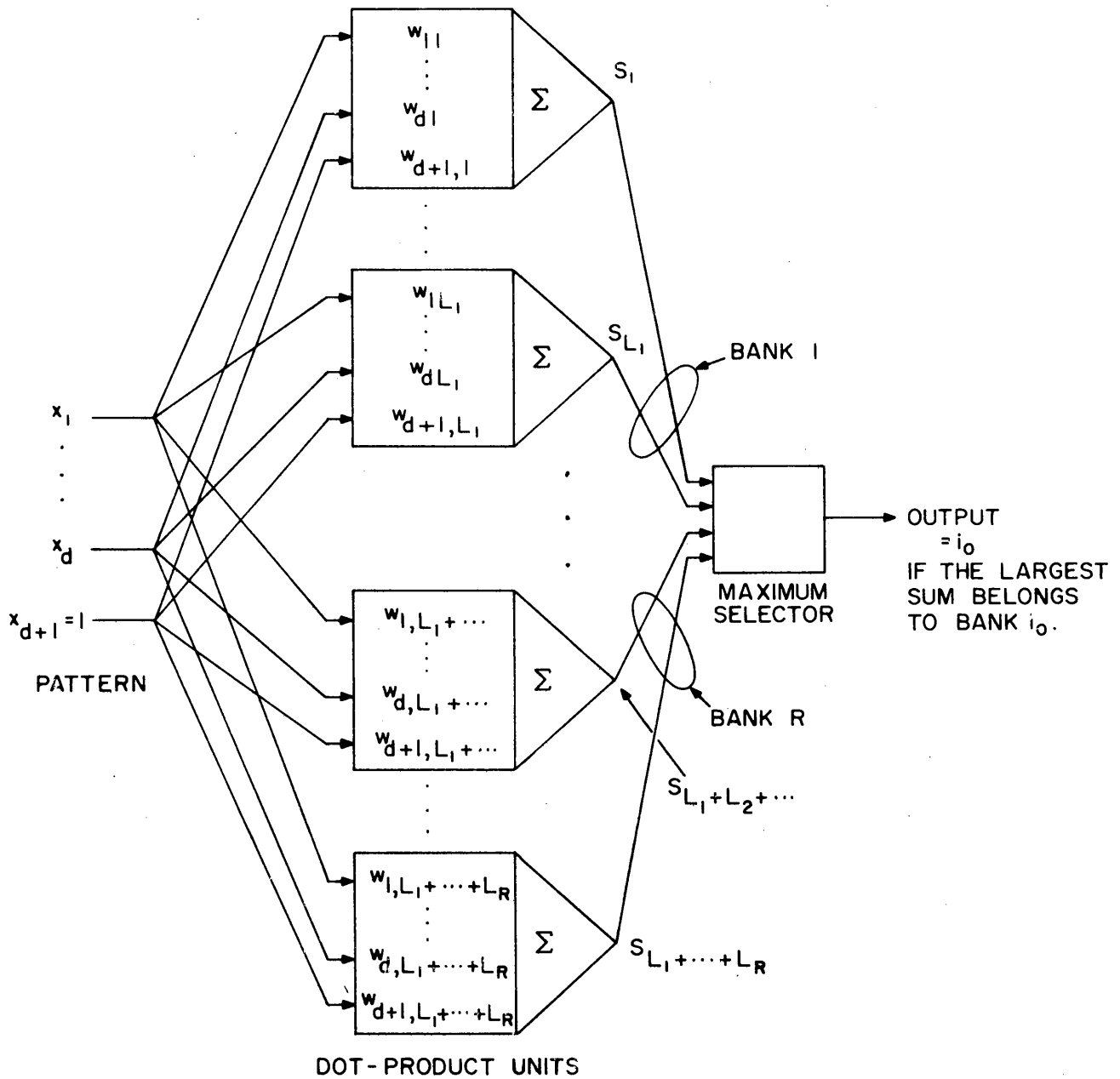


FIG. 4 AN R-CATEGORY PNL MACHINE

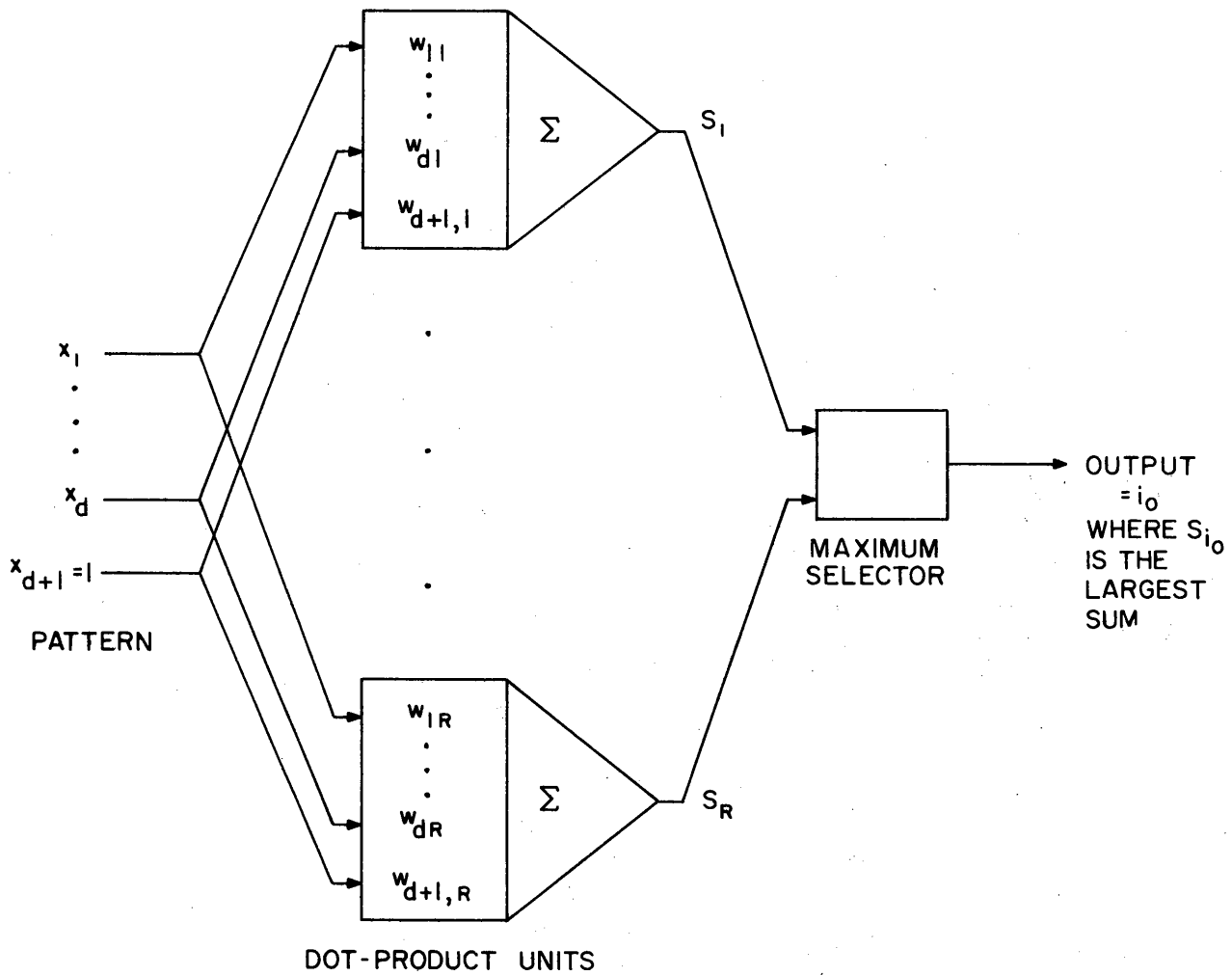


FIG. 5 AN R-CATEGORY LINEAR MACHINE

Various other structures have been suggested for pattern classification; we shall review them briefly here. These other structures all have one common attribute: They consist of parallel organizations of simpler two-category classifiers. We shall call a two-category classifier a dichotomizer.

The simplest dichotomizer is perhaps a linear machine. For $R = 2$, a linear machine consists of two DPU's followed by a maximum selector. The maximum selector determines which DPU output is largest. This determination could be made by comparing the difference of the DPU outputs with a threshold value of zero. But this difference can be computed by a single DPU whose weights are equal to the differences of the weights of the original DPU. Such a structure, a DPU followed by a threshold device, we shall call a threshold logic unit (TLU).

The TLU is illustrated in Fig. 6. The threshold element responds with a +1 if the DPU output is positive, and it responds with a -1 if the DPU output is negative. These two possible outputs are then associated with the two pattern categories.

One can combine several TLU's to form a more complex dichotomizer called a committee machine.^{4,5} A committee machine is a dichotomizer that bases its classification on the majority vote of an odd number of trainable TLU's. Each TLU responds with a +1 or -1 to any pattern, and a "vote-taker" sums the outputs of the TLU's. This sum must be either positive or negative, since there are an odd number of TLU's. If the sum is positive, the committee machine responds with an output of +1; if the sum is negative, the committee machine responds with an output of -1. These two outputs are then associated with the two pattern

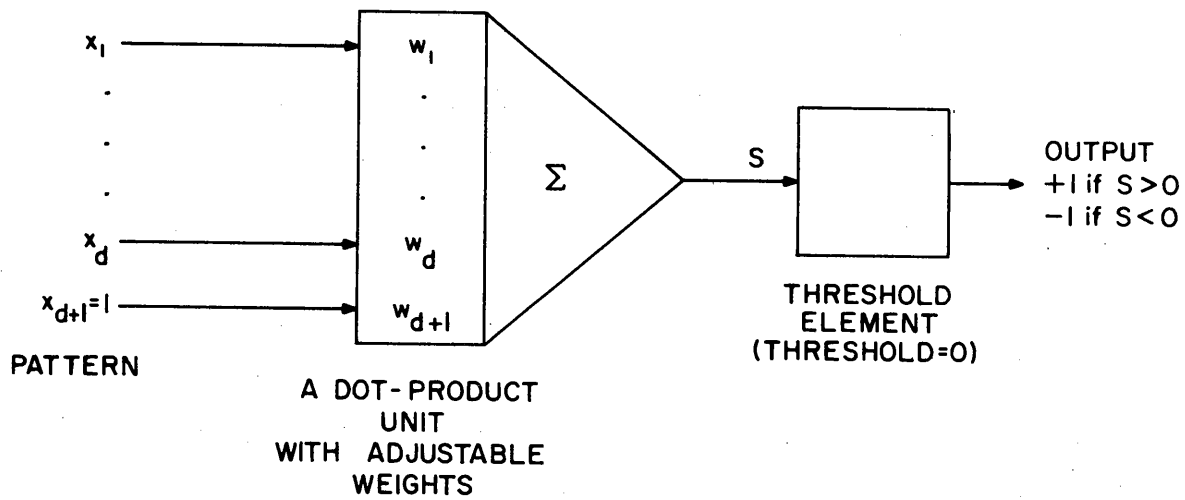


FIG. 6 A THRESHOLD LOGIC UNIT (TLU)

categories. The committee machine is illustrated in Fig. 7. This organization has been used in at least two pattern-recognition devices.^{4,6}

2. Parallel Organization of Dichotomizers

One approach to the design of pattern-classifying systems for more than two categories ($R > 2$) has employed a parallel organization of dichotomizers. The dichotomizers may be of any type; for example, they may be PWL machines, simple TLU's, or committee machines. A classifier employing some number, say q , of independent dichotomizers can classify patterns into as many as 2^q different categories since there are 2^q different possible output combinations. This type of dichotomizer uses a code converter to implement a coding scheme whereby the R pattern categories are associated with the 2^q possible dichotomizer output combinations. Such a classifier is illustrated in Fig. 8.

It has been found useful to use a value of q such that 2^q is much larger than the number of categories R . Such a redundant use of dichotomizers permits the use of error-correcting codes. For example, suppose $q = 9$ and $R = 32$. There exists a single-error-correction nine-bit code with five information bits. For each of the 32 code words belonging to this code, there corresponds one of the pattern categories. This correspondence defines the desired outputs for each of the nine dichotomizers. When a pattern is presented to this machine the response of five of the dichotomizers can be used to specify the 32 categories after any single errors have been corrected in the complete nine-bit output.

Another type of code employs as many dichotomizers as there are categories ($q = R$). Each dichotomizer is a specialist. The i th

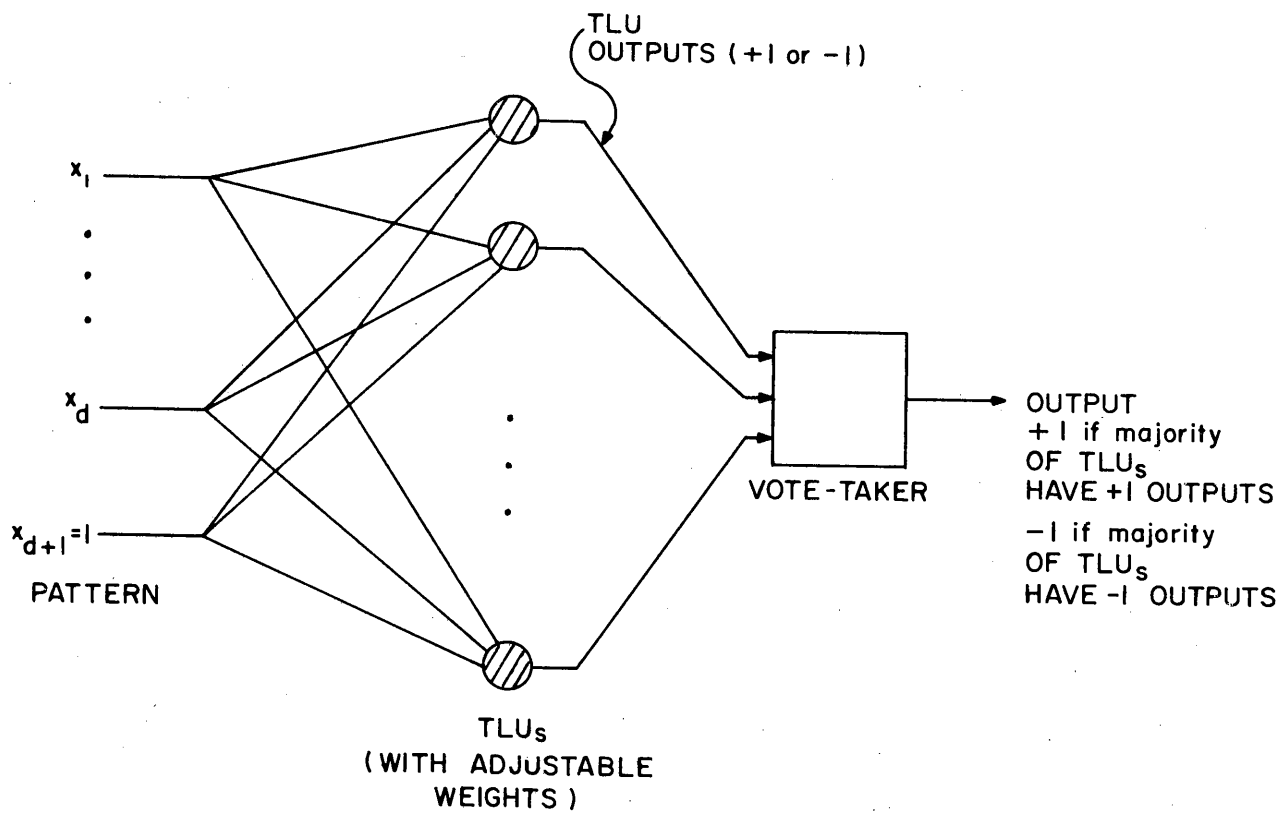
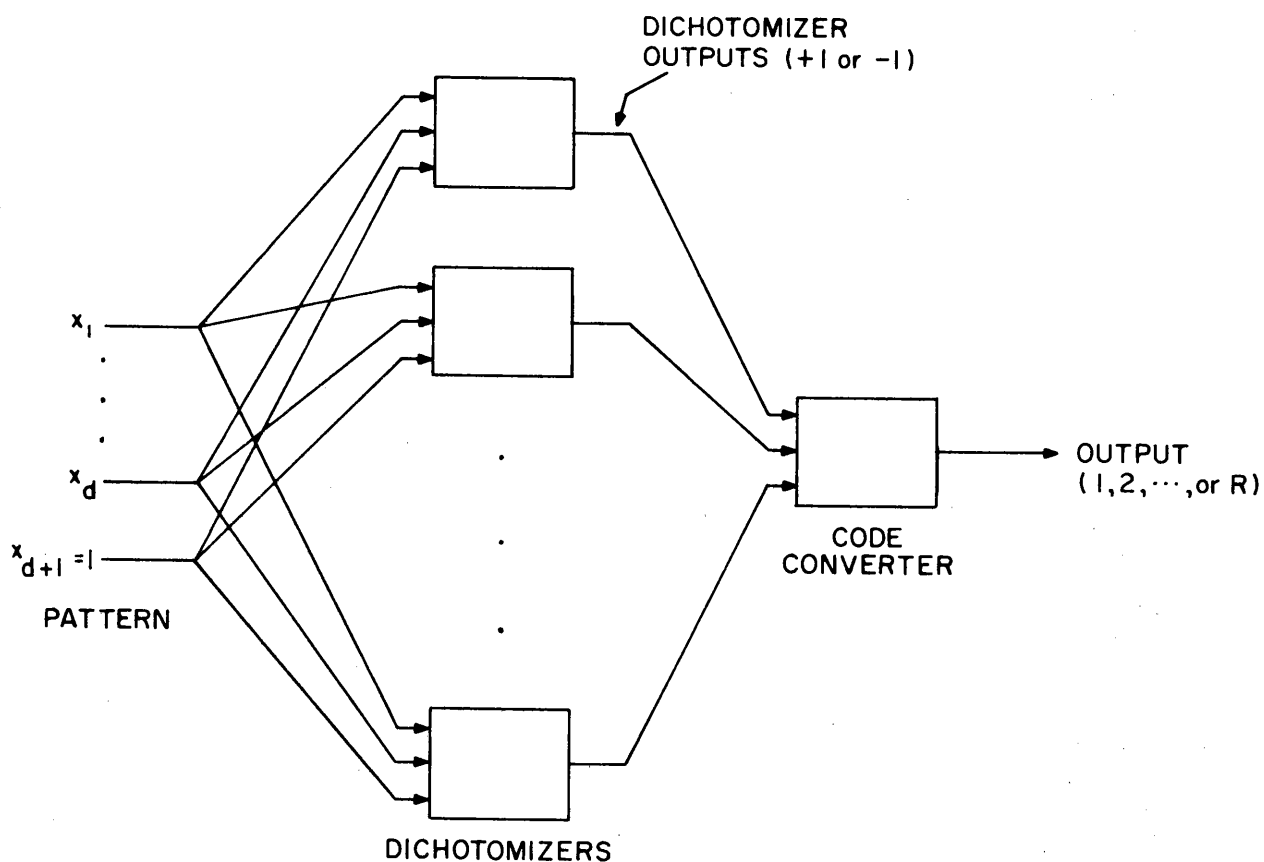


FIG. 7 A COMMITTEE MACHINE



- NOTE: 1. THE DICHOTOMIZERS MAY BE OF ANY TYPE. A SPECIAL CASE OF INTEREST OCCURS WHEN THEY ARE SINGLE TLU_s .
2. THE CODE CONVERTER MAY IMPLEMENT VARIOUS CODING SCHEMES INCLUDING ERROR-CORRECTING AND "SPECIALIST" CODES.

FIG. 8 AN R-CATEGORY CLASSIFIER USING PARALLEL DICHOTOMIZERS

dichotomizer then would respond with a +1 to all patterns in category i and with a -1 to all other patterns. Here we must interpret illegal response combinations (all R dichotomizers responding with a -1, or more than one dichotomizer responding with a +1) as a reject or confused decision.

One of the outstanding research problems connected with the parallel dichotomizer organization is the question: Which coding scheme to use? Very little is yet known, for example, about the trade-off between redundancy in the number of dichotomizers and accuracy of classification.

An important special case of the parallel dichotomizer organization occurs when each dichotomizer consists of a single TLU. In this case it would seem necessary to compensate for the simplicity of the constituent dichotomizers by a more redundant use of them. Thus, 2^q is usually very much larger than R.

Some interesting coding schemes have been proposed to associate the different combinations of TLU response with the pattern categories. One such scheme, employed by a machine organization called Tokoloshe⁷ used $q = 63$ and $R = 64$. There exists a code consisting of 64 code words, each of 63 bits and each code word differing from the other code words in precisely 32 bits. These 64 code words were used to define the desired response of the 63 TLU's for each of the 64 pattern categories. In using such a pattern classifier, the properties of the code permit up to 15 errors in the individual TLU's without incurring a classification error.

Similar codes exist for any q such that $q = 2^p - 1$ where p is any integer. They are called maximal-length shift-register sequences⁸

and have the following properties:

- (1) The code length is $2^p - 1$ bits.
- (2) There are 2^p distinct code words.
- (3) Each code word differs from all the others in precisely 2^{p-1} bits allowing up to $[2^{p-2} - 1]$ errors.

Additional redundancy in the number of TLU dichotomizers can be obtained by using codes employing a higher ratio of word length to number of words. For example, a maximal-length shift-register sequence code with $2^p > R$ could be used. Suppose $p = 6$ and $R = 16$. In this case, the 64 different code words might be assigned to the 16 categories by associating four code words with each category. This extra redundancy permits a larger number of permissible combinations of TLU responses for each category.

Other types of codes might also be used for this type of organization. It has been suggested that it would be appropriate for the code words corresponding to the pattern categories to differ from each other in a manner that reflects the relative importance of each category and the relative costs of errors. Thus, if it is important to avoid confusing Categories i and j , the corresponding code words should differ in a large number of bits.

Except for some of these intuitive suggestions, which experience has not yet raised above the level of doubt, very little is yet known about how best to provide the requisite redundancy of TLU's in such a classifier.

It is interesting to note that a pattern classifier consisting of

parallel committee machine dichotomizers could also be interpreted as a parallel organization of single TLU dichotomizers. This is so because each committee dichotomizer consists of a layer of parallel TLU's. The TLU's from the different dichotomizers can be lumped together and a code specified that would yield the same category decisions as does the combination of committee machines.

3. \mathbb{T} Processors

Each of the pattern classifiers that we have considered so far used the same basic element: a dot-product unit.

If the DPU weight set is represented by the vector \vec{W} and a $(d+1)$ th weight w_{d+1} , then for any pattern \vec{X} , the dot-product unit computes

$$S = \vec{X} \cdot \vec{W} + w_{d+1} \quad (19)$$

The adjustable parameters of this sum (i.e., the components of \vec{W} and w_{d+1}) occur linearly in S , and many of the theoretical results concerned with trainable classifiers depend only on this fact. These results are easily generalized⁹ to machines containing elements that compute weighted sums of functions of the components of \vec{X} .

If the pattern \vec{X} is transformed by some non-adjustable vector-valued function $\vec{F}(\vec{X})$ into a new set of components, these new components can be used as the input to the trainable pattern classifier often with enhanced results. We speak of any device making such a transformation as a \mathbb{T} -processor.

An example of a \mathbb{T} -processor is a device which for any pattern \vec{X} with d components (x_1, x_2, \dots, x_d) produces a set of $\frac{d(d+3)}{2}$ new

components, which are all the products of the x_i taken up to two at a time (i.e., $x_1^2, x_2^2, \dots, x_d^2, x_1x_2, x_1x_3, \dots, x_{d-1}x_d, x_1, x_2, \dots, x_d$). These $\frac{d(d+3)}{2}$ components can then be applied to the dot-product units of a pattern classifier. If the pattern classifier is a TLU, the combination of a \mathbb{T} -processor and a trainable TLU can implement hypersurfaces much more complex than the hyperplane implemented by a TLU alone. For example, when products of up to two of the x_i are computed by a \mathbb{T} -processor, the tandem combination can implement general hyperquadric surfaces that have significantly more power to dichotomize pattern sets than does the simpler hyperplane.

Thus, the \mathbb{T} -processor is a technique to enhance the classification power of relatively simple trainable organizations. Additional research is needed to determine whether or not it competes favorably with the techniques involving the more complex trainable organizations that we have already described.

E. THE PROBLEM OF TRAINING PATTERN-CLASSIFYING MACHINES

Some of the pattern-classifier structures we have discussed were the consequences of what appeared to be reasonable methods of using a training set of patterns to classify other patterns whose categories are unknown. For example, nearest-mode classification is one method of using the training patterns. This method led us to consider the PWL machine structure. We have noted that other structures consisting of parallel combinations of TLU's might also be useful as pattern classifiers. Having compiled a list of interesting classifier structures, we could reconsider as a separate problem the task of training these structures. Training now means adjusting the DPU weights

such that the structure performs acceptably on the patterns in the training set. Our attention during this project has been devoted primarily to the problem of training linear and PWL machines. Some methods for training the other types of structures have been considered previously in other projects.*

At least two approaches can guide the training of linear and PWL machines. One of these is to adjust the weight vectors in such a way that they seek centers of high pattern density. We shall call this type of training mode-seeking training. Another approach is to present patterns to the machine in an iterative manner and make correcting adjustments in the weight vectors if and only if the machine misclassifies a pattern. We shall call this type of training error-correction training. The weight vectors that result from error-correction training will not necessarily have obvious relationships to modes of the pattern distribution since the training process terminates whenever all of the patterns in the training set are classified correctly.

Both approaches to training a linear machine are fairly well understood. Mode-seeking training of a linear machine is generally only appropriate when there is only one mode per category. In this case these modes are usually well estimated by the center of gravity of the patterns in each category.

Error correction training methods for linear machines are known which are guaranteed to be successful whenever the patterns in the training sets can be perfectly classified by a linear machine. These methods

* See Ref. 6 and 10 for a discussion of committee machine training and Ref. 7 for a discussion of parallel TLU training.

are discussed in detail in the next section.

The problem of training a PWL machine is more complex, and our knowledge is incomplete. Mode-seeking training is difficult because most procedures for estimating the modes of multimodal distributions from samples are cumbersome. Some error-correction training methods have been suggested and tested, and these are generally much simpler to apply than the mode-seeking methods. We shall discuss the status of both of these training methods for PWL machines in detail in the next section.

II TRAINING METHODS

A. LINEAR MACHINES

We have mentioned two training methods for linear and PWL machines: mode-seeking and error-correction training. When mode-seeking training is used to train a linear machine, the tacit assumption of unimodality is made. That is we assume that each category consists of a single cluster of patterns. The center of gravity of each cluster would then serve as a mode or "typical pattern" for each category. Computing the center of gravity of a set of patterns is a straightforward matter and needs no further comment here. Therefore we shall proceed immediately to a discussion of error correction training methods.

Several error-correction training methods¹¹ are known for a special case of the linear machine, the TLU. Recently, these methods have been generalized¹¹ to linear machines with $R > 2$. We shall discuss here one method, called the fixed-increment training rule.

A linear machine for $R > 2$ was illustrated in Fig. 5. It consists of R DPU's and a maximum selector. The outputs of the R DPU's are given by the expressions:

$$S_i = \vec{W}_i \cdot \vec{X} + w_{d+1,i} \quad (20)$$

for $i = 1, \dots, R$

The vectors \vec{W}_i are called weight vectors. The components of the i th weight vector are given by $w_{1i}, w_{2i}, \dots, w_{di}$. The $(d+1)$ th weight $w_{d+1,i}$ provides a bias term that is independent of the pattern.

We assume that we are given a fixed, finite, linearly separable set of training patterns. That is, we assume that there exists a linear machine, called a solution machine, that correctly classifies all of the training patterns. We desire to train the linear machine by adjusting its weights so that it responds correctly to every pattern in the training set. A response to a pattern in category i is correct only if the output of the i th DPU is the largest.

The fixed-increment error correction procedure is guaranteed to find a set of solution weights when they exist. In this procedure, each pattern in the training set is presented one at a time in any sequence. Arbitrary initial weights are selected for the machine and adjustments of these are made only when the machine responds incorrectly to any training pattern. Suppose that a pattern, \vec{X} , belonging to category i , is presented with the result that some DPU, say the j th ($j \neq i$) has an output sum larger than that of the i th. That is, the machine erroneously places \vec{X} in category j . The weights used by both the i th and the j th DPU's will then be modified.

Let the weights of the i th and j th DPU's prior to modification be denoted by

i th DPU

\vec{w}_i and $w_{d+1,i}$

j th DPU

\vec{w}_j and $w_{d+1,j}$

If the same weights after adjustment are denoted by primes, $(')$ the adjustment rule can then be written as follows:

$$\vec{W}_i' = \vec{W}_i + a\vec{X}$$

$$w_{d+1,i}' = w_{d+1,i} + a$$

and

$$\vec{W}_j' = \vec{W}_j - a\vec{X} \quad (21)$$

$$w_{d+1,j}' = w_{d+1,j} - a$$

where a , a positive constant, is called the correction increment. The effect of the adjustment is to increase the output of the i th DPU while decreasing the output of the j th DPU. It has been proven¹¹ that this error correction procedure, using any positive value of a , is guaranteed to find a set of solution weights after only a finite number of adjustments if such a set exists. Observe that for $R = 2$ this error correction rule is identical with the well-known rule for training a TLU.

The above rule has been studied during this project by Duda who has developed more general conditions for weight modification that are sufficient to guarantee convergence. These conditions allow modifications to the weights other than those given by Eq. (21) but include those of Eq. (21) as a special case. Duda's general conditions on the training rule are stated, and convergence is proven in an appendix to this report.

B. PIECEWISE LINEAR MACHINES

1. Mode-seeking Training

a. Stark's method

The problem of locating centers of high pattern density has not yet been solved in an entirely satisfactory manner. Some iterative

methods have been proposed and we have conducted some experiments with these. Many of these experiments have been with two-dimensional patterns, because the results can then be easily visualized and displayed. In this section we shall give a brief exposition of some of these mode-seeking methods and illustrate them with a description of our experiments with two-dimensional patterns.

The first mode-seeking method with which we have experimented is a relatively simple one adapted from a procedure originally suggested by Stark, Okajima and Whipple¹². We shall call it Stark's method here, although we realize that it probably has since been modified and improved.

Stark's method is an iterative procedure for adjusting weight vectors such that, finally, each weight vector is the center of gravity of all of the patterns closest to it. Let the initial weight vectors be selected arbitrarily.* We arrange the training patterns into a training sequence and examine them one at a time. We shall describe what happens upon examination of the k th pattern in the training sequence. Suppose this pattern is denoted by \vec{X}_k . We now look for that weight vector that is closest (Euclidean distance) to \vec{X}_k . Suppose at the k th stage the j th weight vector, denoted by $\vec{W}_j[k]$, is closest[†] to \vec{X}_k . Then, only this closest weight vector is adjusted and all other weight vectors are left fixed.

*One possible selection is to set each of them equal to a different training pattern.

[†]Ties are resolved by an arbitrary rule such as: Select that weight vector whose index is smallest.

The adjustment made to this closest weight vector is to move it part way along a line directed toward the pattern \vec{X}_k .

The adjusted weight vector $\vec{W}_j[k+1]$ is then given by the following expression:

$$\vec{W}_j[k+1] = \frac{N \vec{W}_j[k] + \vec{X}_k}{N+1}, \quad (22)$$

where N is the number of times that this weight vector had been adjusted before. Thus, the change in the weight vector is calculated to treat equally all of the patterns affecting it. Other averages could be used; in particular, it may be advantageous to weight recent patterns more strongly than patterns occurring earlier in the training sequence.

The reader is invited to test the above rule graphically using simple clusters of two-dimensional patterns. Each pattern is considered in sequence and the closest weight vector is moved toward that pattern. After many steps, the actual motions become smaller and smaller at each step due to the averaging process. Eventually the weight vectors settle down at locations that are centers of gravity of subsets of the patterns; it is hoped that each of these final subsets corresponds to a cluster of patterns.

We shall illustrate Stark's method by an experiment using 225 two-dimensional patterns. These patterns are plotted as points in Fig. 9. These points are grouped in 9 clusters of 25 points each; each cluster is numbered as in Fig. 9. One point in each cluster is taken to be the "center" of the cluster. These center points are:

2.0, 8.0

4.0, 9.0

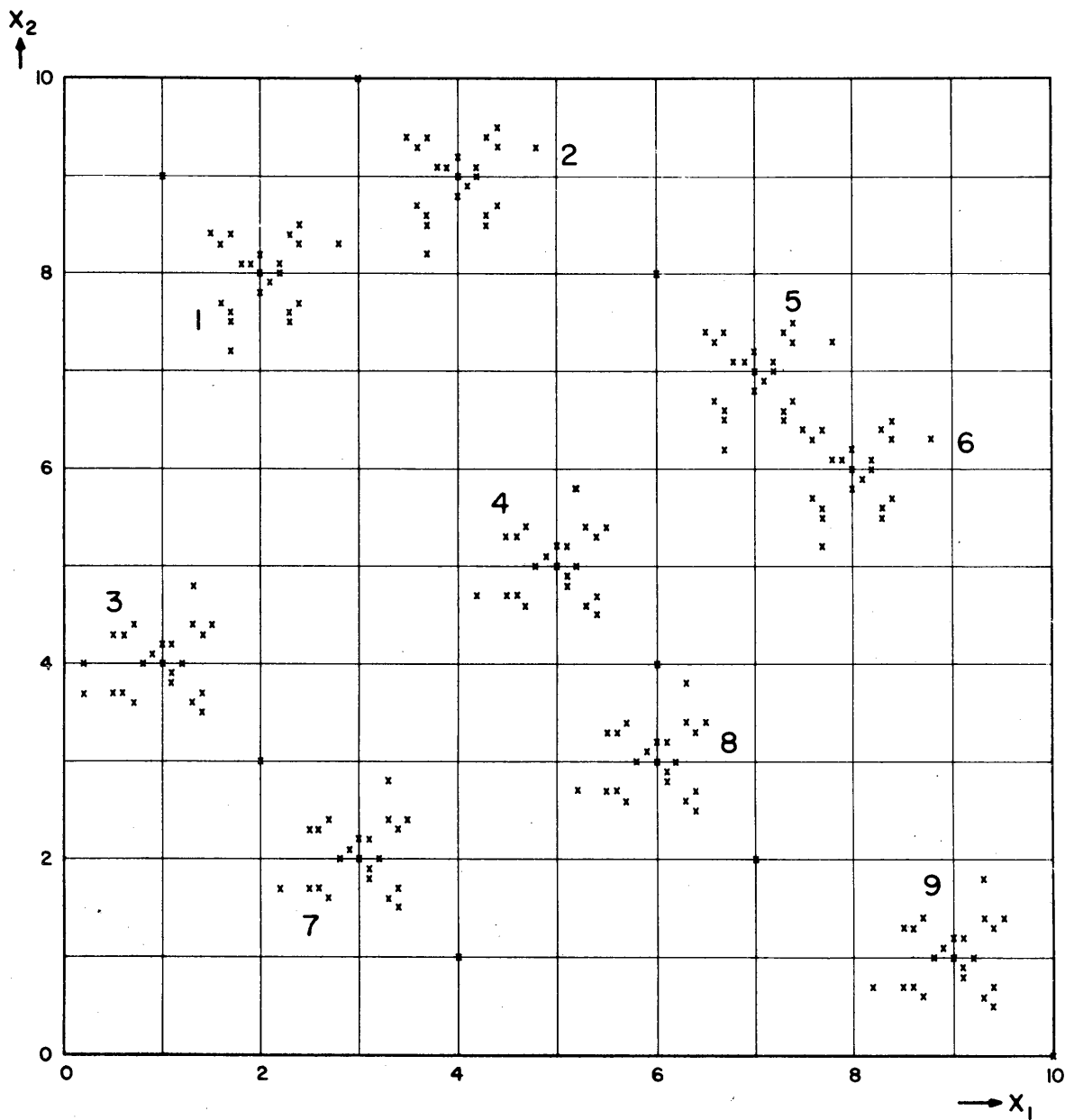


FIG. 9 PATTERNS FOR MODE-SEEKING EXPERIMENTS

1.0, 4.0
 5.0, 5.0
 7.0, 7.0
 8.0, 6.0
 3.0, 2.0
 6.0, 3.0
 9.0, 1.0

The other points in each cluster were chosen to appear as if they were drawn from bivariate normal distributions with $\sigma = 0.5$. Cluster membership is obvious for most of the patterns in Fig. 9 although the members of Clusters 5 and 6 tend to intermingle. Our test of Stark's method began with nine initial weight vectors all residing at the point 5.0,5.0.

The set of 225 patterns was arranged in random order, the first few patterns were:

(91,12), (94, 13), (10, 40), (81, 59), (61, 28), (58, 30), (43, 86), etc.

After one cycle through the list of 225 patterns the weight vectors occupied the following positions (to four decimal places):

After 1 cycle

<u>Weight Vector</u>	<u>Residing in Cluster Number</u>
9.240, 0.9800	9
1.9837, 3.0204	between 3 and 7
7.4780, 6.5240	between 5 and 6
5.9481, 2.9444	8
2.9800, 8.5240	between 1 and 2
5.3200, 5.4200	4

After 1 cycle (cont'd)

<u>Weight Vector</u>	<u>Residing in Cluster Number</u>
5.2000, 4.7875	4
4.7000, 5.1375	4
4.6000, 4.6667	4

Two more cycles through the patterns did not disturb the general locations of the weight vectors. In fact only the last two of the nine weight vectors were moved at all during these two cycles and these only slightly. The last two weight vectors on the list now occupied the following positions:

4.7455, 5.1773

4.5273, 4.6727

The locations of all nine weight vectors at the end of the third cycle are marked by small squares (□) in Fig. 10.

A verbal discussion of the results of this experiment will help to reveal some of the limitations of Stark's method. Of the nine mode-seeking weight vectors, two located the centers of Clusters 8 and 9, respectively. Three others located the centers of cluster pairs. These cluster pairs were 1-2, 5-6, and 3-7. The four remaining weight vectors remained close to their origin within Cluster 4. These four weight vectors each migrated to the center of a "subcluster" within Cluster 4. Thus, our experiment with Stark's method revealed two types of "misbehavior":

- (1) Some weight vectors stabilized in intercluster space at the center of-gravity of pairs of clusters
- (2) Some weight vectors stabilized at the center of gravity of a subset of a cluster.

Referring to Fig. 10, it would have been encouraging had one of the Clusters 3 or 7 attracted one of the "extra" weight vectors from within Cluster 4 so

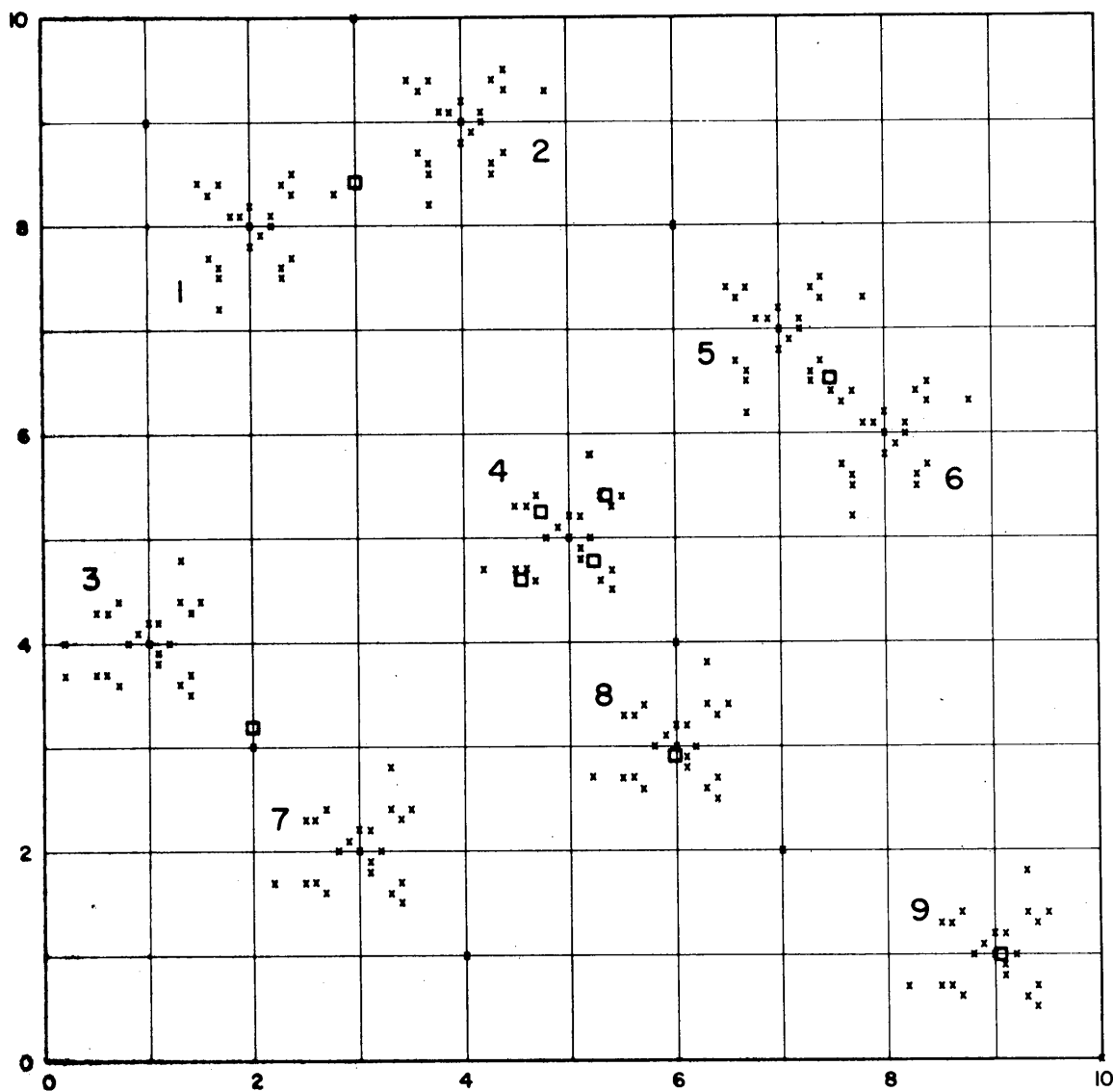


FIG. 10 NINE WEIGHT VECTORS FOUND USING STARK'S METHOD

that Clusters 3 and 4 would have each had a weight vector. But the single weight vector lying between Clusters 3 and 7 is already closer to each of the patterns in these two clusters than are any of the weight vectors in Cluster 4. Therefore conditions allowed Stark's process to stabilize before each cluster had its own weight vector.

Since we can envision circumstances in which we would desire that every cluster have its own weight vector, we would like other mode-seeking methods that are more apt to ensure that result. Perhaps what is needed is to modify Stark's method by adding some extra features. One useful feature might be to lump together weight vectors which are "too close" (thus eliminating all but one of the weight vectors within Cluster 4). Another useful feature might be to split weight vectors lying at the center of clusters which are "too extensive" (thus creating a pair of weight vectors to handle Clusters 3 and 7, for example). We shall see in the next section that a process developed by Ball and Hall¹³ at SRI called ISODATA incorporates just these features.

b. ISODATA

G. Ball and D. Hall of SRI have developed a mode-seeking process called ISODATA, which is quite closely related to Stark's method. Although ISODATA was not actually developed as a modification of Stark's method, it is easier to describe it as if it were.

Suppose we first modify Stark's method by saving all weight vector modifications until the end of each cycle through the pattern set. Let the initial weight vectors occupy arbitrary but different positions. Then each cycle consists first of determining for each weight vector that subset

of patterns which are closer to that weight vector than to the other weight vectors. Each weight vector is then moved to the center of gravity of its corresponding subset. So far the process is the same as Stark's method except the weight adjustments are made in response to the complete set of training patterns simultaneously rather than in response to one pattern at a time. After moving the weight vectors, the pattern subsets closest to each relocated-weight vector are re-established, their centers of gravity computed, and the weight vectors moved again. This process continues for as many steps as might be required. Each step is called a cycle.

In ISODATA there are also provisions for splitting, combining and discarding weight vectors. These three operations can occur if appropriate after every relocation of the weight vectors or more infrequently.

Discarding weight vectors is subject to the following discard test: If the number of patterns closest to a given weight vector falls below a critical threshold θ_N , then that weight vector is discarded.

Combining weight vectors is subject to the following combining test: If the Euclidean distance between two weight vectors falls below a critical distance, θ_C , then these two weight vectors are combined in a new weight vector located at the center of gravity of the combined associated subsets of patterns.

Splitting a weight vector is subject to the following splitting test: The marginal standard deviations (the standard deviations of each of the d components) are computed for each subset of the patterns. If for any pattern subset a marginal standard deviation is larger than a critical value,

θ_E , then the weight vector at the center of gravity of that subset is split into two weight vectors. The components of the two new weight vectors are the same as those of the original weight vector except for that component corresponding to the maximum standard deviation. To the value of this component is added a positive increment (small compared to the average pattern component value) for one of the new weight vectors and a negative increment of the same size for the other new weight vector.

To avoid combining weight vectors that have just been split, it is convenient to adopt the conventions: Allow splitting at the end of odd-numbered cycles only and allow combining after even-numbered cycles only.

With the addition of these features, marked mode-seeking improvement over Stark's simple method is seen. An experiment using ISODATA on the pattern set illustrated in Fig. 9 was conducted using five initial weight vectors. These five weight vectors, denoted by circles, are illustrated in Fig. 11. The discard, combining and splitting parameters for this experiment were:

$$\theta_N = 10, \quad \theta_C = 1.0, \quad \theta_E = 0.5$$

The result of the first cycle is depicted in Fig. 11. The subsets of patterns closest to each of the original five weight vectors are separated by the line segments shown. Weight Vectors 1, 3 and 4 are moved to the center of gravity positions marked by squares (\square). Weight Vectors 2 and 5 are also moved and then subsequently split, since for these pattern subsets θ_E was exceeded and the first cycle is an odd-numbered one. (θ_E was also exceeded for the pattern subset surrounding Weight Vector 3, but Weight

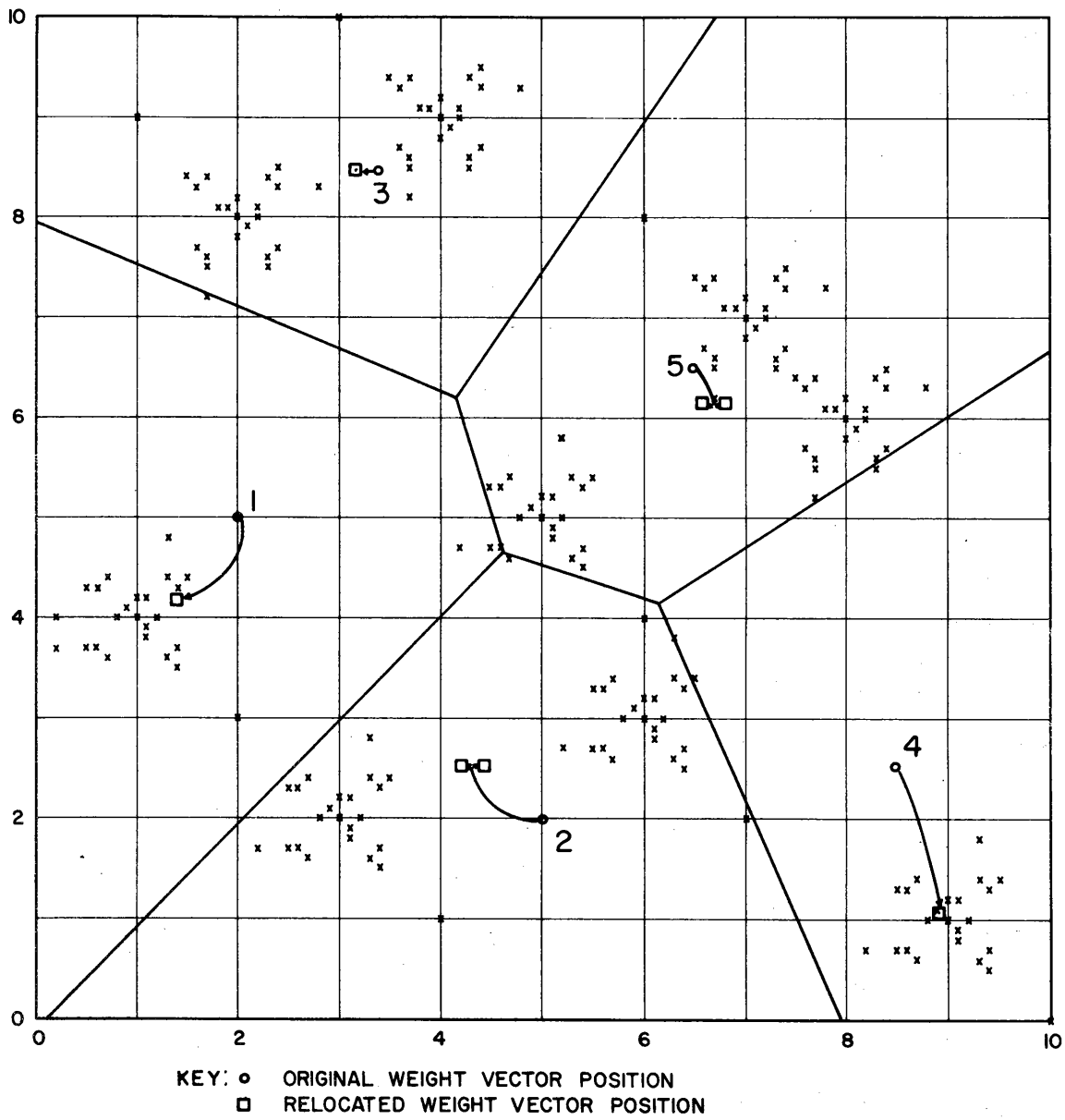


FIG. 11 RESULT OF FIRST CYCLE OF ISODATA PROCESS

Vector 3 was not split at this time because during this particular experiment an added condition was required before splitting. This condition was that the average distance from the center of gravity of a subset of patterns to all patterns in the subset had to exceed this same quantity averaged over all subsets.)

The result of the next cycle of the process is shown in Fig. 12. Here the weight vectors are moved to their new center of gravity positions, but no splitting is allowed because this is an even-numbered cycle. During the third cycle, illustrated in Fig. 13, Weight Vectors 3 and 5 are split. There are now nine weight vectors, one for each cluster. The final result after four cycles of ISODATA is shown in Fig. 14. Here the nine weight vectors are each in the center of a cluster, and the mode-seeking process has been successfully completed.

The provisions for generating new weight vectors and discarding others is an important feature of ISODATA. Yet these provisions require specification of the process parameters θ_N , θ_E , and θ_C . If these parameters were selected inappropriately in our experiment just described, the process might not have stabilized with weight vectors at the centers of all nine clusters. For example, selection of the splitting parameter, θ_E , is critical. In our experiment θ_E was set at 0.5. The patterns in each cluster were initially chosen so that the standard deviation would be approximately 0.5 (the actual standard deviations were slightly less than 0.5). If θ_E were set much lower than 0.5, more splitting would have occurred and some pattern clusters would have had two or more weight vectors.

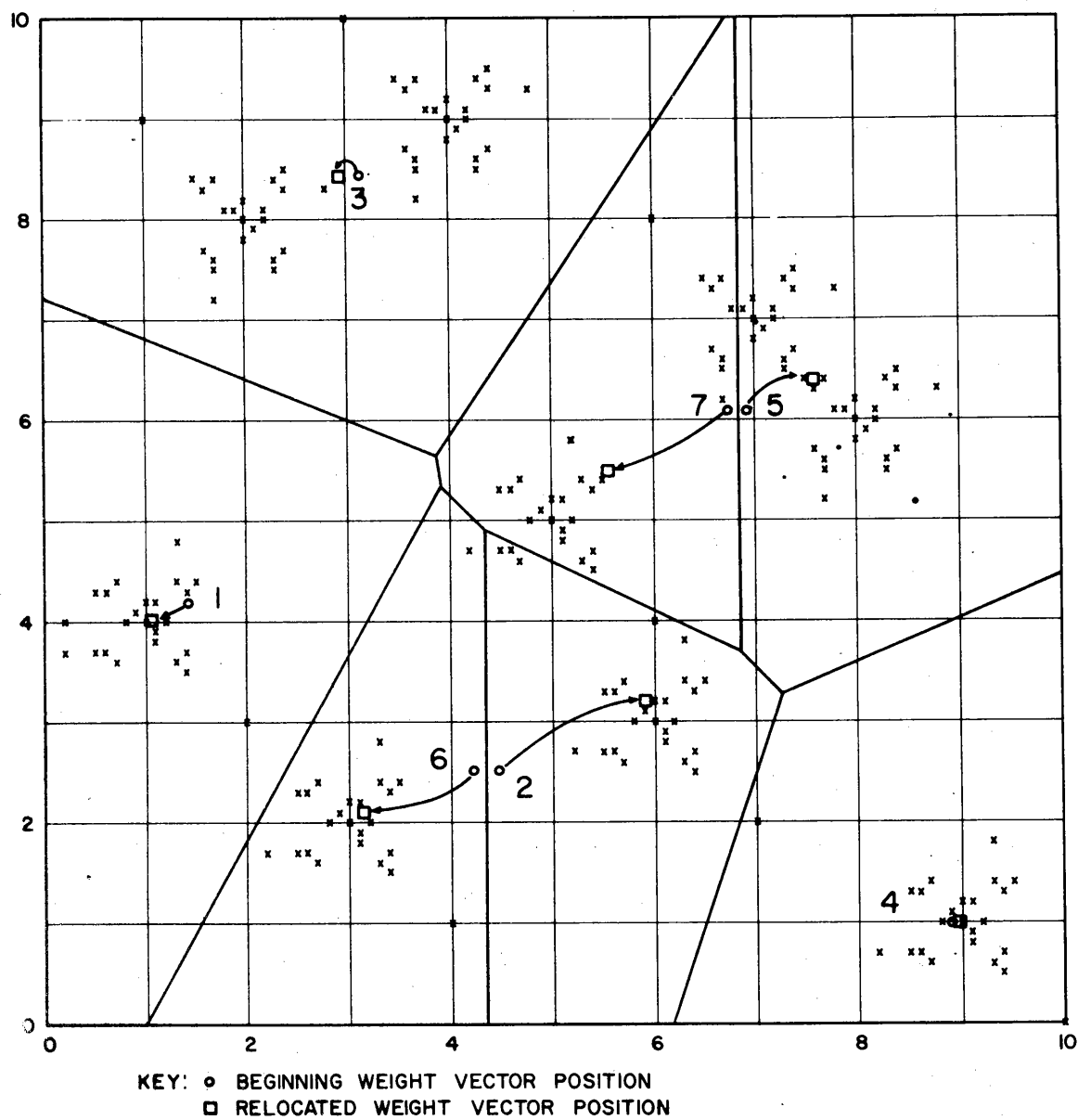


FIG. 12 RESULT OF SECOND CYCLE OF ISODATA PROCESS

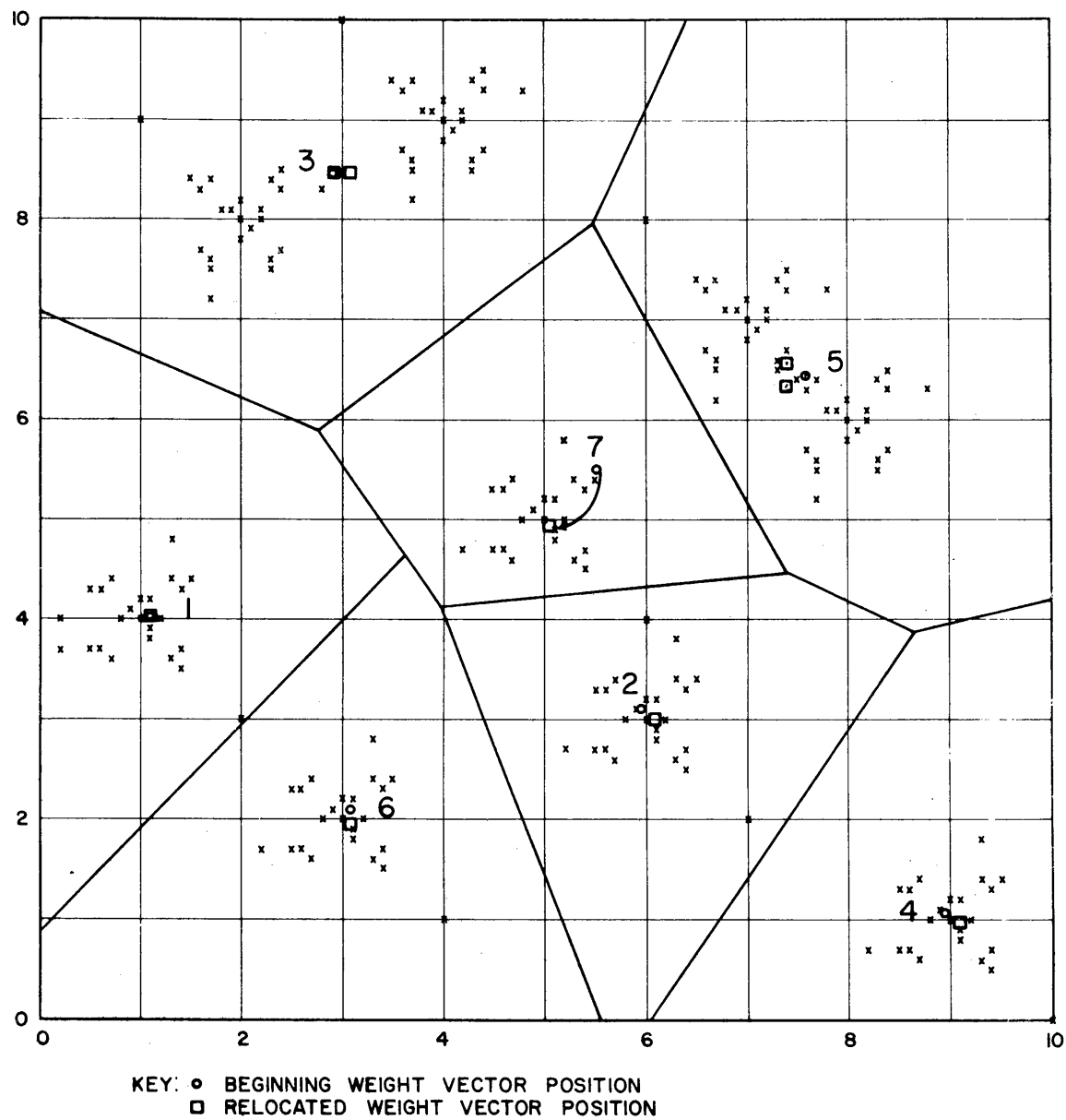


FIG. 13 RESULT OF THIRD CYCLE OF ISODATA PROCESS

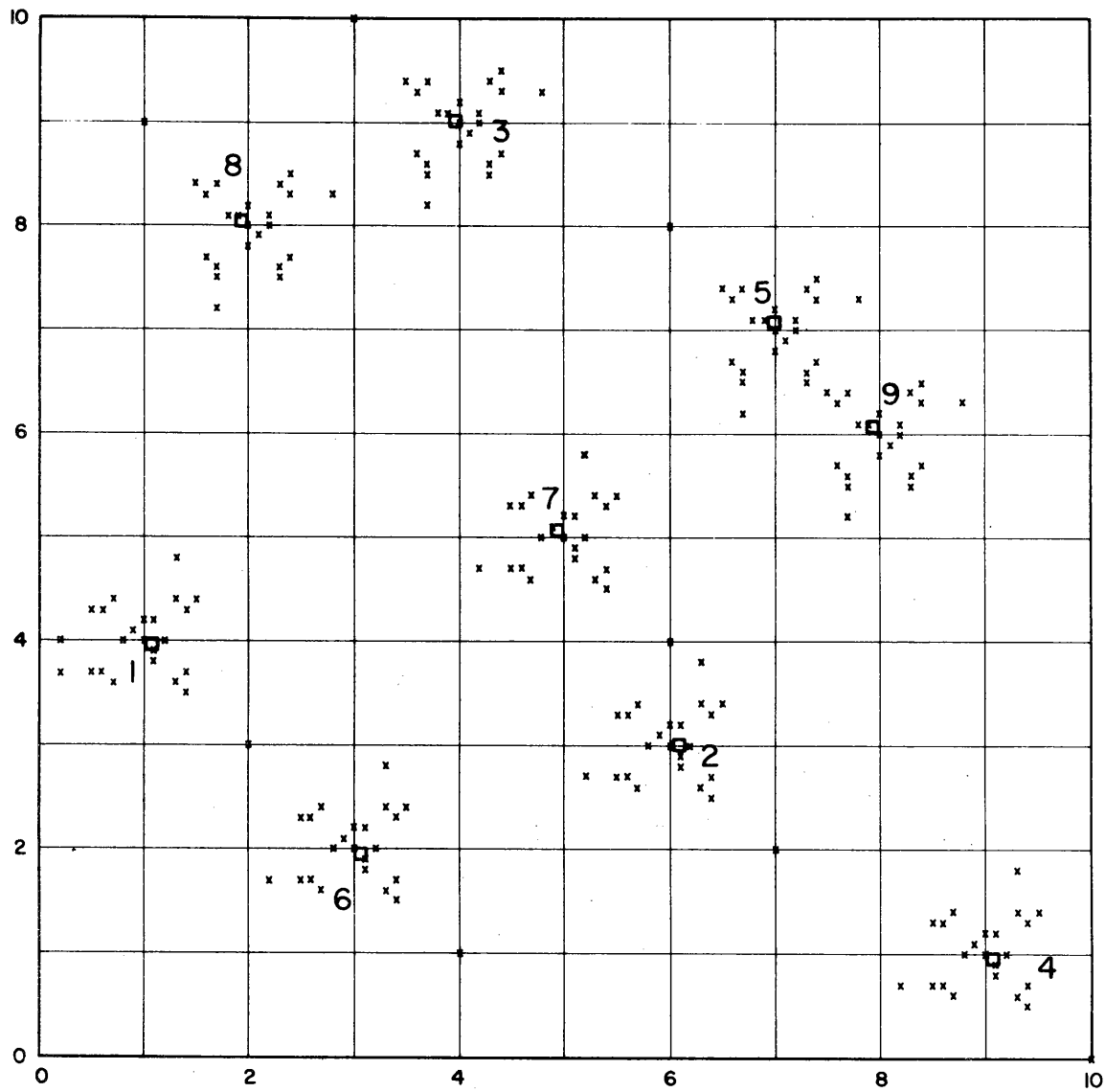


FIG. 14 FINAL RESULT AFTER FOUR CYCLES OF ISODATA PROCESS

If θ_E were set much larger than 0.5, clusters that were close together (e.g., Clusters 5 and 6) would probably have had to share a single weight vector. One improvement to ISODATA would be a provision for weight vector birth that does not depend so critically on the selection of some process parameter.

We have omitted from our discussion of Stark's method and ISODATA one important aspect of the pattern-classification problem. Usually we have available the knowledge of the categories of the training patterns; yet, the patterns shown in Fig. 9 were not divided into category sets. One way in which these mode-seeking methods could take into account category information is to repeat the mode-seeking process for each category of patterns. Thus the patterns in Fig. 9 might be regarded as all belonging to the same category, say Category 1. Another group of patterns might be the training patterns for Category 2; an independent process might operate simultaneously to find the centers of clusters of the Category 2 patterns, etc. In this way the centers of all clusters of patterns would be found, and these "modes" could be used as the weight vectors of a closest-mode classifying machine.

This suggestion for the utilization of category information suffers from one serious drawback, however. Clusters of patterns belonging to different categories should exert interdependent (rather than independent) influence on the final positions of the weight vectors. To illustrate why this should be so, consider the final positions of the weight vectors in Fig. 14. Suppose all of the patterns in that figure are regarded as belonging to Category 1. Then Weight Vectors 5 and 9, for example, could be

combined into one weight vector with no increase in error rate for a closest mode classifier if the Category 2 patterns closest to this combined 5-9 mode lie at some remote distance. Thus, a process that considers all patterns of all categories simultaneously could lead to a closest-mode classifier requiring fewer weight vectors than the total number of clusters. This process must be capable of finding all of the clusters, however, if the closest mode-classifier requires them in order to perform optimal classification.

A mode-seeking method will be described in the next section, which requires fewer arbitrary process parameters than ISODATA and also uses the pattern category information in an interesting way.

c. Rosen-Hall Method

An interesting mode-seeking method utilizing pattern-category information has been suggested by C. Rosen and D. Hall of SRI during this project. This process, which we call the Rosen-Hall method, operates in the following way: An initial weight vector is selected to represent each category. Thus, for an R-category problem, we begin with R weight vectors. These initial weight vectors can be conveniently established by selecting at random one training pattern from each category and setting the initial weight vectors equal to these patterns.

We next determine the subsets of training patterns closest to each weight vector as in the ISODATA process. Now, however, each pattern in a subset that does not belong to the same category as the weight vector of that subset is put on an ERROR list. The ERROR list will sometimes have patterns on it belonging to all R categories.

The patterns on the ERROR list are now grouped according to category. If more than N_1 patterns of a given category, say Category i , occur on the ERROR list then a new weight vector is established for Category i . The new weight vector is set equal to that pattern in Category i on the ERROR list which is farthest (Euclidean distance) from the original Category i weight vector. Thus after establishing an ERROR list we may establish some new weight vectors equal to some of the patterns on the ERROR list.

We next move each original weight vector, if it qualifies, to the center of gravity of the subset of patterns closest to it, excluding those patterns on the ERROR list. A weight vector does not qualify for motion if the number of patterns in its subset is smaller than N_2 . If a weight vector does not qualify, it is discarded. None of the weight vectors just created from the ERROR list are either moved or discarded during this cycle. The first cycle of the process is now complete.

The second and subsequent cycles of the process are the same as the first with the exception that what are regarded as original weight vectors at the beginning of any cycle are all the weight vectors surviving and generated during the previous cycle. New weight vectors are now set up to be those patterns on the ERROR list farthest from any existing like-category weight vector.

The Rosen-Hall method was applied to the patterns in Fig. 9. The process parameters during this experiment were $N_1 = 5$ and $N_2 = 10$. This time, however, the patterns of Fig. 9 were grouped into five categories. These patterns and the category groupings are shown in Fig. 15. Note that the Category 2 and Category 3 patterns are unimodal, the Category 1 and Category 5 patterns are bi-modal, and the Category 4 patterns are tri-modal.

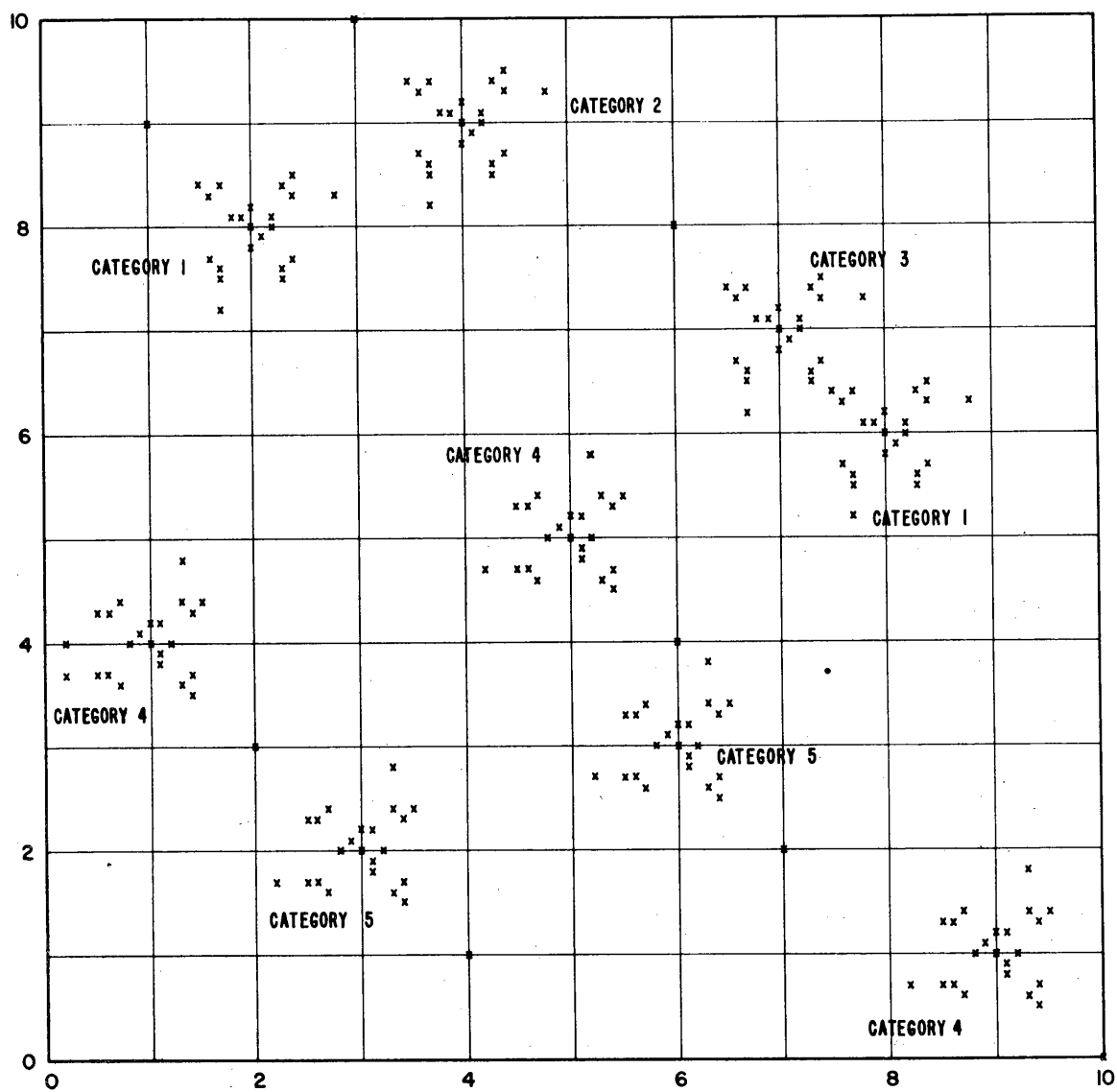


FIG. 15 PATTERN CATEGORIES FOR ROSEN-HALL METHOD EXPERIMENT

We begin with five weight vectors: These are shown in Fig. 16 where they are depicted as circles. The subsets of patterns closest to each of the original weight vectors are separated by the line segments shown. There are 83 patterns on the ERROR list; they include some Category 1, Category 4 and Category 5 patterns. Three new weight vectors are created, one for each of these three categories represented on the ERROR list. These new weight vectors are illustrated as triangles (Δ) in Fig. 16. All five of the original weight vectors qualify for motion, and these are moved to the center of gravity positions marked with a square (\square) in Fig. 16.

Progress during the second cycle is shown in Fig. 17. One of the Category 5 weight vectors is discarded. There are 68 patterns on the ERROR list, and these lead to the creation of two new weight vectors. The intermingling of Category 1 and Category 3 patterns from adjacent clusters always causes some Category 1 and/or Category 3 patterns on the ERROR list. These errors would have led to the creation of unnecessary new weight vectors had N_1 been too low.

After the third cycle, each of the nine clusters has a weight vector at its center. These are illustrated in Fig. 18. After the third cycle, the situation stabilizes with a total of only two patterns on the ERROR list. These are the patterns: (6.0, 4.0), a member of Category 4 but closer to a Category 5 weight vector; and (7.0, 7.0), a member of Category 1 but coincident with a Category 3 weight vector. Thus a closest-mode classifier using the weight vectors established by the Rosen-Hall method would make only two errors on the training patterns, less than 1 percent.

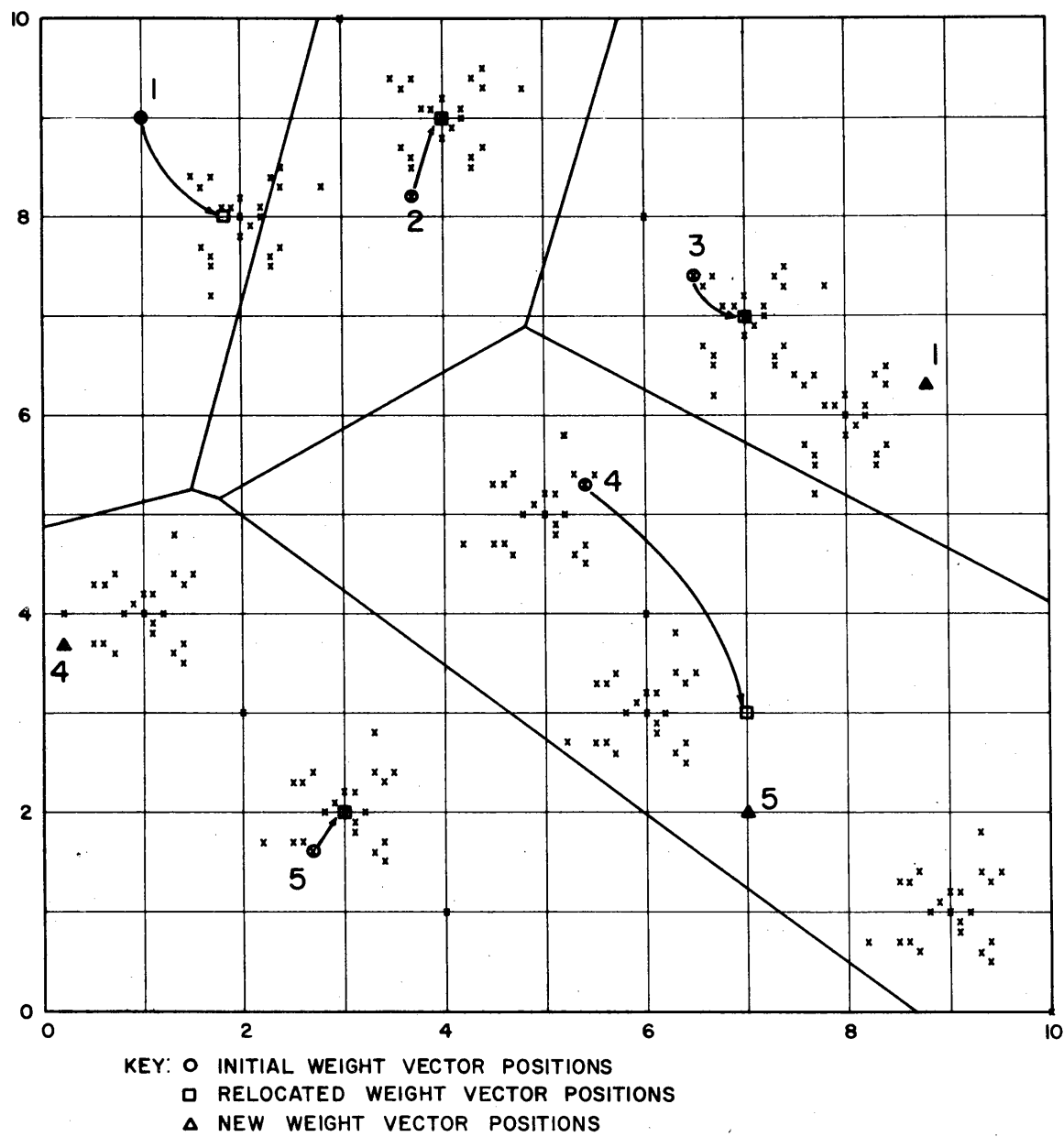


FIG. 16 RESULT OF FIRST CYCLE OF ROSEN-HALL METHOD

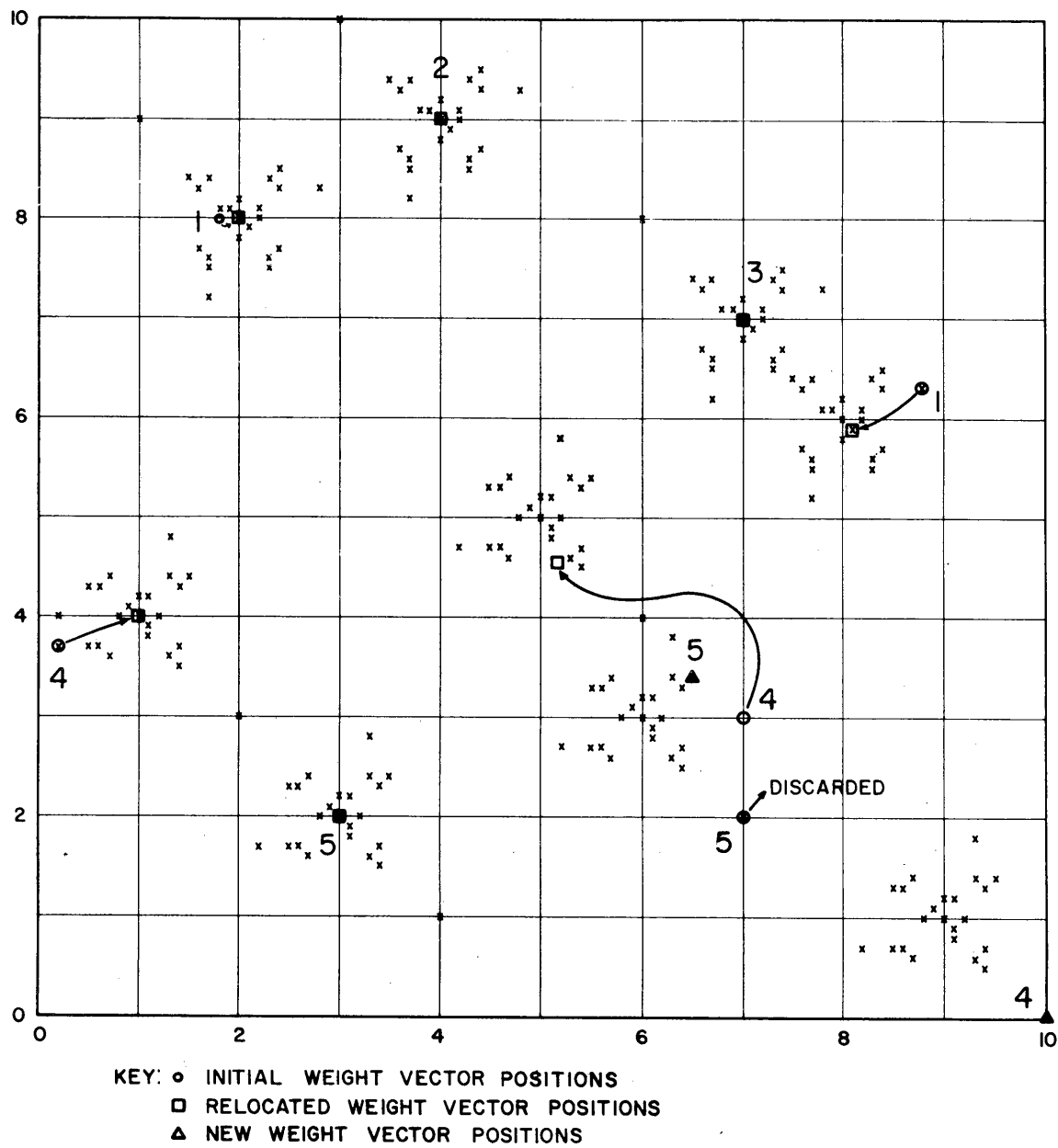


FIG. 17 RESULT OF SECOND CYCLE OF ROSEN-HALL METHOD

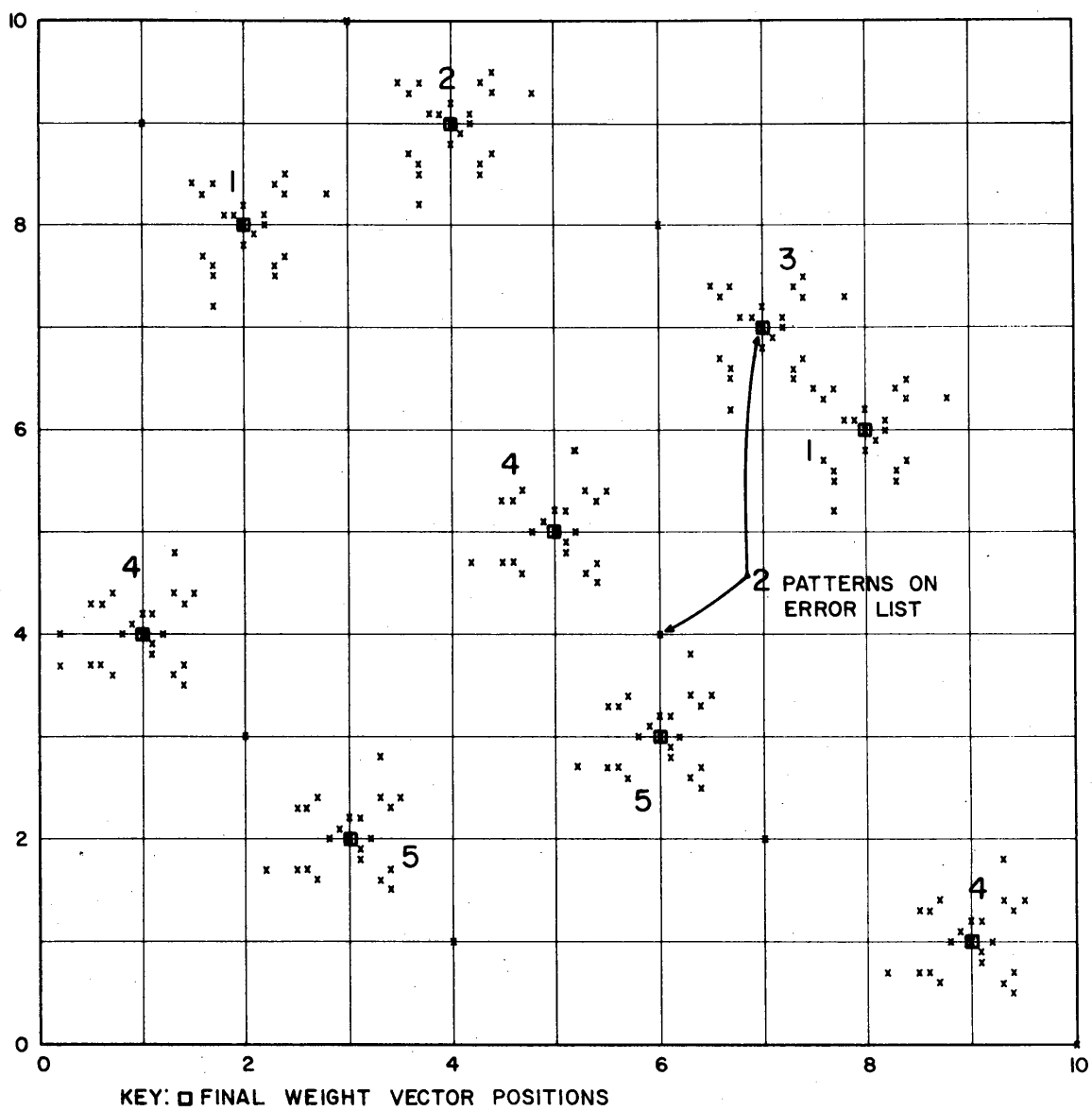


FIG. 18 FINAL POSITIONS OF WEIGHT VECTORS
AFTER THIRD CYCLE OF ROSEN-HALL METHOD

The above experiment is representative of the few that have been conducted using the Rosen-Hall method. This method appears to have many advantages over the other methods described for mode seeking. It has fewer process parameters than ISODATA; the two that it does have are easily set. Typically N_1 should be set to reflect the expected percentage error of the classifier on the training patterns, but its exact value is not too critical. The value of N_2 should depend on the size of the training set and the number of weight vectors available. For example, if we had 5000 training patterns, we might reasonably insist on having at least 100 patterns in a cluster if we had around 50 weight vectors available. Perhaps in this case we might let N_2 be somewhat lower than 100, say 50.

The Rosen-Hall method can also be modified to handle a pattern at a time in iterative fashion. In this modification, yet to be tested, we would present a pattern at a time and select initial weight vectors in some arbitrary fashion. Suppose a pattern belonging to Category i is presented. We would first determine the closest weight vector. If the closest weight vector is a Category i weight vector we would move it a short distance toward the pattern. The distance would be calculated so that each weight vector would always be a running average of all patterns toward which it moved. If the closest weight vector was not a Category i weight vector we would not move any of the weight vectors but would instead put the pattern on the ERROR list. This process would continue until the ERROR list grew large enough, at which time new weight vectors would be established as before.

2. Error-Correction Training

a. The Two Aspects of Training PWL Machines

An alternative to mode-seeking training of PWL machines is

error-correction training. We have seen that there do exist error-correction training methods for linear machines that are guaranteed to terminate in zero-error solutions if such exist. No error-correction methods for PWL machines exist that carry the same guarantee, however. Some error-correction adjustment procedures have been suggested that have proved useful in a variety of experiments, but unfortunately these methods do not always lead to a zero-error solution when it exists. Some of these experiments will be described later, but first we shall discuss in more detail the problems of training a PWL machine.

The problem of error-correction training of a PWL machine has two aspects. First, the weight vectors themselves must be adjusted in response to classification errors. Second, the weight vectors must be organized into R banks. This organization should be regarded as a training problem since it might be unknown beforehand how many weight vectors should be in each bank. Thus, training also involves shuffling the weight vectors from bank to bank (i.e., from category to category) until some appropriate organization is found.

What is needed, then, to train PWL machines is a method of adjusting weight vectors and a method for transferring them from bank to bank. Preferably, these training procedures should be iterative so that complicated analyses of large pattern sets might be avoided. The simplest, and perhaps most desirable, procedure would involve forming a training sequence of patterns and presenting these patterns, one at a time, to the machine. At each pattern presentation the machine may be adjusted by changing weight

vectors, by transferring weight vectors among the banks, or both.

The second aspect of training might be avoided by ensuring a surplus of weight vectors in each bank, although such an exuberant use of weight vectors might be grossly uneconomical in terms of equipment and training time. If the total number of weight vectors were limited, it might then become important to be able to move them from banks where they weren't needed to banks where they were needed.

No completely satisfactory iterative methods have yet been proposed for transferring weight vectors among the banks,* although later we shall discuss some preliminary experiments tested with some proposed methods.

b. An Error-Correction Method for Adjusting
Weight Vectors in a PWL Machine.

The following procedure was suggested by Nilsson¹⁴ for training a PWL Machine. It is an error-correction method which adjusts weight vectors but leaves their distributions within the banks fixed. In application of this method then, some arbitrary number of weight vectors must be assigned to each bank.

*Perhaps it would be useful to regard the matter of transferring weight vectors among the banks in terms of a birth and death process and a pool of "extra" weight vectors. Whenever a weight vector is found to be superfluous within a given bank, it should be transferred from the bank to the pool (death). Whenever a bank needs an extra weight vector, one should be transferred to that bank from the pool--provided that the pool is not empty (birth). What is needed now are rules defining when a weight vector within a bank is no longer needed there and when a new weight vector is needed within a bank.

Training patterns are presented to the PWL machine one at a time for trial. After presenting a pattern that the machine classifies correctly, we make no changes in any of the weight vectors. Suppose, however, that a pattern, \vec{X} belonging to Category i causes an incorrect response. Such would be the case if the j th bank, $j \neq i$, contained the DPU with the largest output. The adjustment method first subtracts \vec{X} from the weight vector used by this DPU in the j th bank. Of those DPU's in the i th bank, we next determine which has the largest output for \vec{X} . The corresponding weight vector is adjusted by the addition of \vec{X} . The intuitive basis for such a procedure is clear; it is an attempt to apply something like the linear machine error-correction procedure to a structure containing more than one DPU per bank.

We shall illustrate the application of this error-correction procedure using the 225 two-dimensional patterns grouped into five categories as shown in Fig. 15. Our illustration experiment will assign as many DPU's to each bank as there are clusters of patterns in the corresponding category. (In experiments to be described later we did not always assign the "correct" number of dot-product units per bank). In this experiment, the PWL machine had two DPU's in Bank 1, one DPU in Bank 2, one DPU in Bank 3, 3 DPU's in Bank 4, and 2 DPU's in Bank 5. This PWL machine is illustrated in Fig. 19.

Before beginning the experiments, the patterns in Fig. 15 were first translated by subtracting 5.0 from each pattern component. This translation shifted the entire pattern set so that it was centered about the origin of the two-dimensional space rather than lying completely within the first quadrant. Experience with the PWL machine error-correction rule

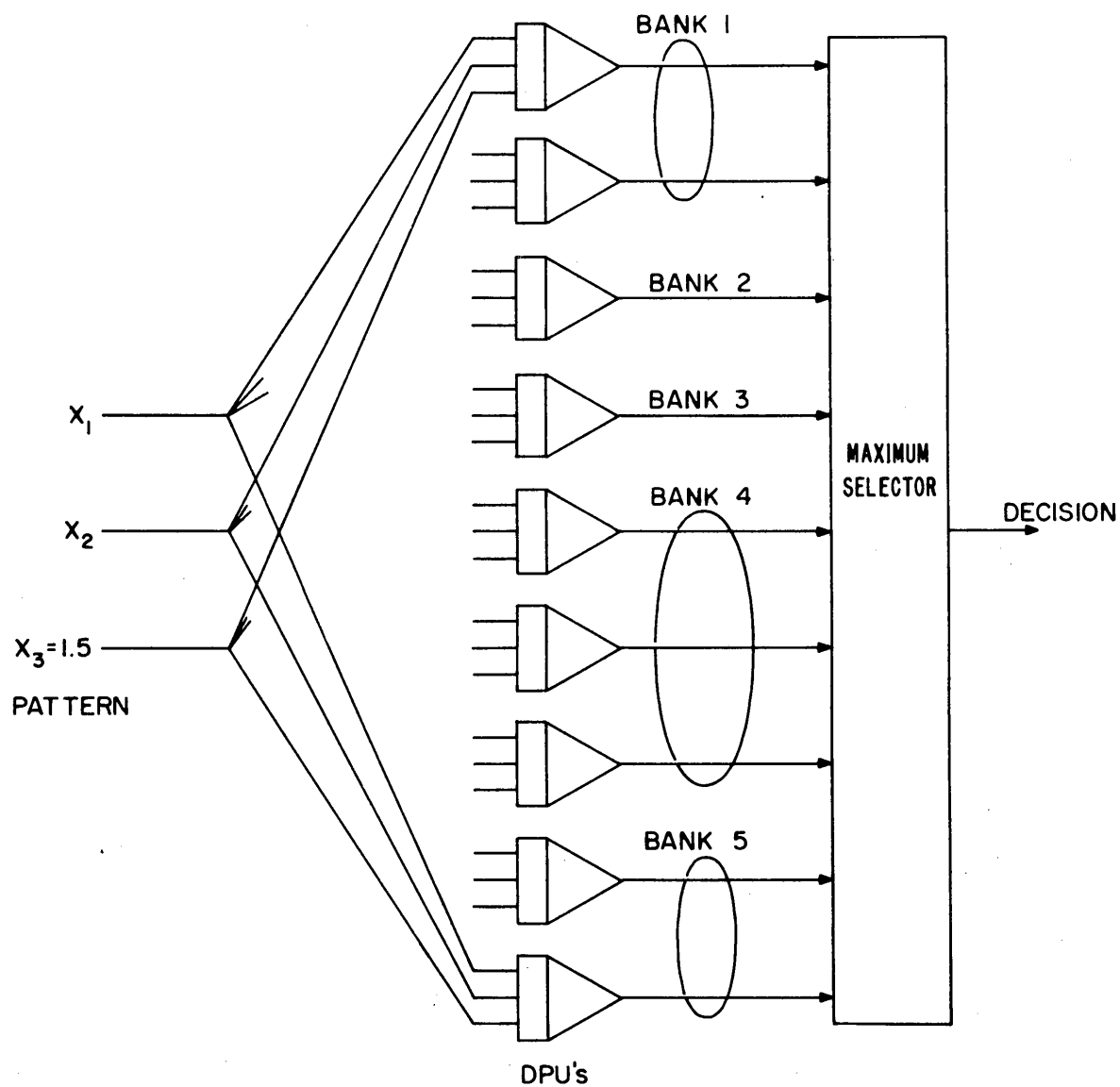


FIG. 19 PWL MACHINE USED IN ERROR-CORRECTION TRAINING
EXPERIMENT WITH TWO-DIMENSIONAL PATTERNS

has shown that such centering of the patterns is necessary to prevent some of the first-adjusted DPU's from being permanently frozen out of action. (If all of the patterns had positive components, some DPU's will soon have each weight negative, leading to perpetually negative dot-products. Then those DPU's with positive components will be the only ones ever to be adjusted.)

The modified patterns were randomly ordered and the resulting list of patterns was presented to the PWL machine cyclicly. The machine was trained using the error-correction rule described above and beginning with all weights set initially equal to zero. The history of the training process is summarized in Fig. 20. Notice that the number of corrections required during a cycle diminishes rapidly with each cycle. During the 16th cycle only four corrections were needed. These corrections and those of the next two cycles result in a set of weights requiring only two corrections during the 19th cycle. It is interesting to examine the decision surfaces at the end of the 19th cycle. The weights obtained after this cycle lead to decision surfaces separating the pattern categories which are quite good indeed. These are shown in Fig. 21. It can be seen by this figure that the trained PWL machine using these decision surfaces will make only 1 error on the 225 training patterns. This pattern is one of the two with components (2.0, 2.0). One member of this pair of identical patterns belongs to Category 1 and the other to Category 3, leading to an inevitable error. This inevitability of error keeps the error-correction process continuing indefinitely although the error rate remains small. If additional patterns, clustered near the training patterns, were classified

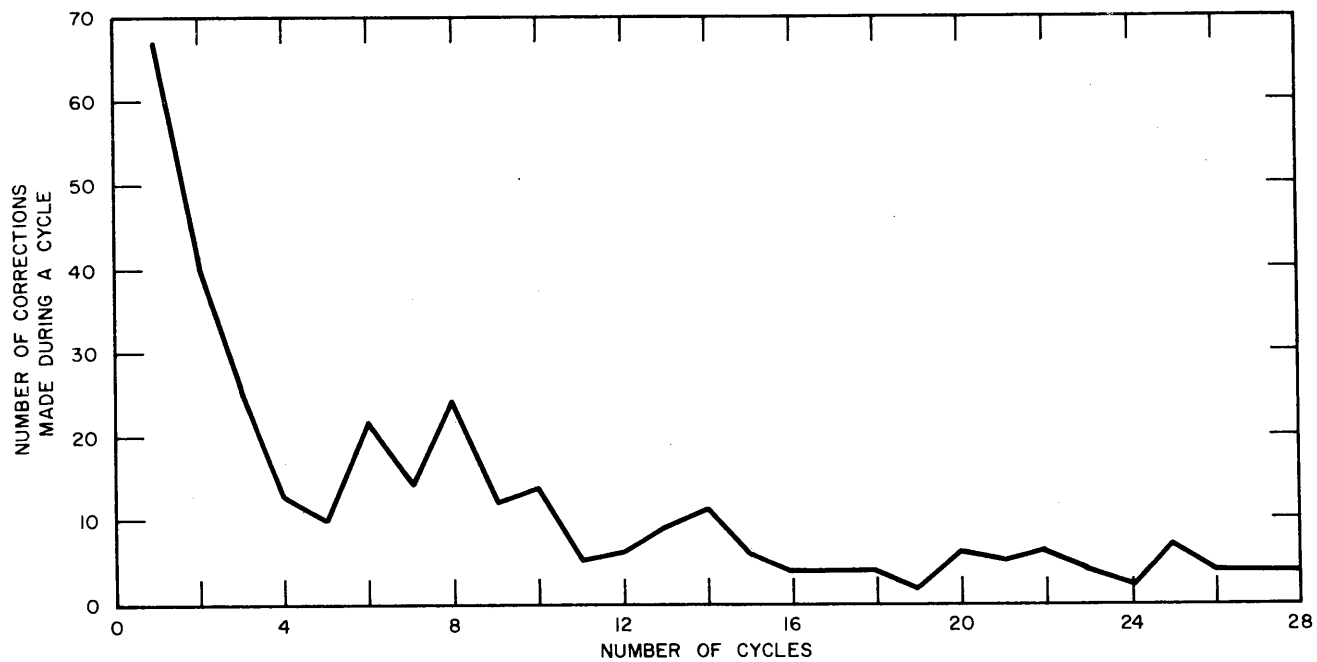


FIG. 20 PLOT OF CORRECTIONS vs CYCLE NUMBER
FOR ERROR-CORRECTION TRAINING OF A PWL MACHINE

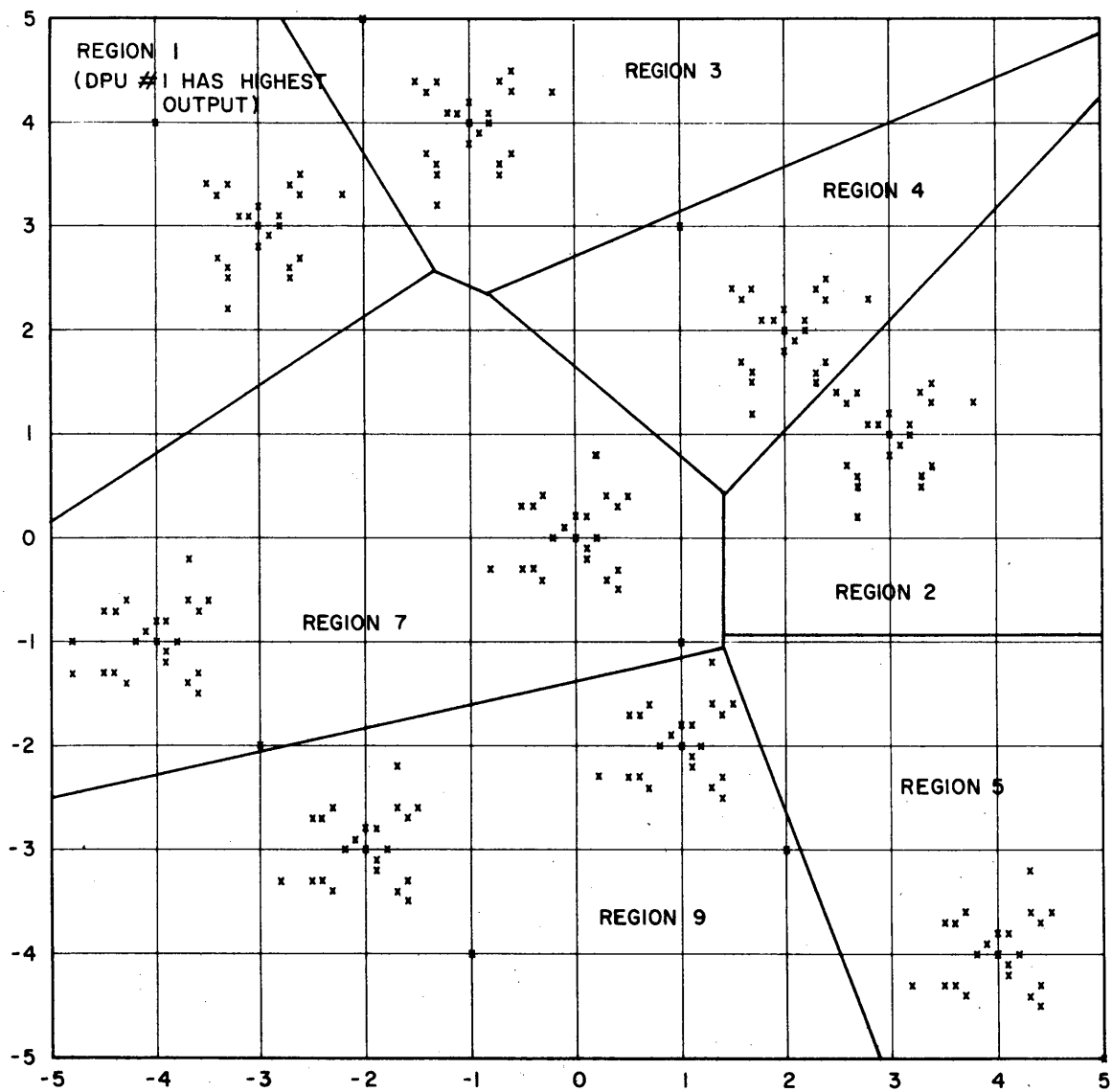


FIG. 21 DECISION SURFACES AFTER 19TH CYCLE

according to the decision regions of Fig. 21, we could expect that these patterns also would be classified with very few errors.

In Fig. 21 we have labeled each region according to which DPU has the highest output for patterns in that region. Note that two of the DPU's (6 and 8) never have the largest outputs for any patterns and thus can be ejected from the trained PWL machine. This fact leads us to predict that we would have been able to train a PWL machine with only seven DPU's to perform as well, even though there are nine clusters of patterns. The PWL machine is often able to group clusters of the same category into a single region with a consequent saving in required DPU's. (In our experiment, two clusters of Category 4 patterns and the two clusters of Category 5 patterns were grouped together.)

The values of the weights used by the PWL machine (Fig. 19) implementing the decision surfaces of Fig. 21 are as follows:

<u>DPU Number</u>	<u>w_1</u>	<u>w_2</u>	<u>w_3</u>	
1	-21.8	9.3	-4.5	} Bank 1
2	22.1	-3.7	-7.5	
3	-0.3	22.1	-7.5	Bank 2
4	2.8	14.7	6.0	Bank 3
5	21.1	-15.3	-15.0	} Bank 4
6	0.1	0.7	-1.5	
7	-13.1	-3.3	25.5	} Bank 5
8	-2.9	1.8	0.0	
9	-8.0	-26.3	4.5	

During this experiment, the w_3 weights multiplied an input component, x_3 , always equal to 1.5. The choice of a value for this (d+1)th input is sometimes critical. It has been found experimentally that a reasonable choice is often the average value of the other pattern components.

C. TRANSFERRING WEIGHT VECTORS AMONG BANKS DURING TRAINING

Some preliminary experimenting with plausible rules for weight vector transferring has been completed during this project. Our experiments were inspired by the following empirically determined factors:

- (1) When a bank contains more DPU's than needed to achieve correct classification of the training patterns, then often one or more DPU's within this bank never contributes the highest dot product.
- (2) When a bank, say the ith, contains fewer DPU's than needed, then during training either there will be a large number of Category i training patterns classified in error or the ith bank will often have the highest dot product for non-Category i patterns.

Our experiments typically began with some arbitrary number of DPU's in each bank. These DPU's were trained by the error-correction training process and were periodically checked for the occurrence of one or the other of the above two factors. At these periods, if Factor (1) were present, a DPU was removed from the appropriate bank; if Factor (2) were present, a DPU was added to the appropriate bank. Two questions arise in connection with adding a DPU to a bank. First one must decide how many errors (of each kind) to tolerate in connection with Factor (2) before deciding to add a DPU. Second, having decided to add a DPU, one must decide on the initial value of its

weight vector.

Our experiments so far have been inconclusive. In some of these we have been able to modify a PWL machine with a poor initial DPU distribution into a PWL machine that correctly classifies all of the training patterns. Other experiments have been unsuccessful in this regard. It is to be hoped that future experimenting will evolve a satisfactory PWL machine training rule covering both aspects of training: weight vector adjustment and DPU transferral.

III ERROR-CORRECTION TRAINING EXPERIMENTS

A. PURPOSE OF THE EXPERIMENTS

In the previous section we described several methods for training linear and PWL machines and illustrated the operation of these methods by applying them to two-dimensional training patterns. The question remains how efficient these methods are when applied to high dimensional patterns. In this section, we shall describe experiments in which linear and PWL machines were trained on 10-dimensional analog and 100-dimensional binary patterns using the error-correction procedure. Mode-seeking training using high dimensional patterns has not been tested during this project, although some experiments on ISODATA have been independently performed by Ball and Hall.¹³ We shall also present experimental results comparing the PWL machine and a network of committee machines trained by error-correction methods on the same training patterns.

In the two-dimensional experiments described in the last section, we could estimate visually the quality of the performance of a trained machine on classifying patterns similar but not identical to those in the training set. The performance of a trainable pattern classifier on an independent test set of patterns is of crucial importance and necessitates explicit experimental investigation for higher dimensional patterns. Of course, in these experiments we are still interested in the speed of training and in the effects of various arbitrary training rule parameters on both training speed and performance on independent test patterns.

In order to evaluate the adequacy of performance of the trained machine on independent test patterns, careful attention was given to the problem of selecting training and testing patterns. If the patterns are governed by completely known probability distributions, a trainable pattern classifier can be evaluated by comparing its performance with the optimum performance. Therefore, it was decided to use artificially generated patterns for which the components were normally distributed and for which the optimum performance was known. In the majority of the experiments the distributions for patterns in the same category were multimodal, the patterns being drawn from unimodal subclasses. The properties of these sets of Gaussian patterns will be described in the next section.

B. THE TEN-DIMENSIONAL GAUSSIAN PATTERNS

The artificially generated patterns were obtained by adding Gaussian random noise to 80 ten-dimensional prototype vectors. The components of these vectors (derived by scaling the first 80 rows of a table of random digits¹⁵) were odd integers from -99 to 99, and the resulting prototype vectors were approximately uniformly distributed throughout a ten-dimensional hypercube centered at the origin.

Four different types of patterns were generated: unimodal spherical, unimodal ellipsoidal, multimodal spherical, and multimodal ellipsoidal. The unimodal patterns were divided into 80 categories, the i th prototype vector being a mean vector for patterns in the i th category. The multimodal patterns resulted from combining these 80 categories as subclasses in a 32-category problem. Eight of the resulting categories were unimodal, eight were bimodal, eight were trimodal, and eight were quadrimodal, the exact subclass

assignments being shown in Table I. In all cases, enough noisy sample patterns were generated to represent all of the modes equally. This means that the a priori probabilities $p(i)$ were equal for the unimodal data, but had a 4:1 range for the multimodal data.

Sample patterns for each category were generated by adding zero-mean Gaussian noise to the components of the prototype vectors. For the spherical data, the noise components were statistically independent with a common standard deviation, σ . For the unimodal spherical data, it is easy to evaluate Eq (1) and obtain an optimal classifier. The optimum performance, of course, depends on σ , and its variation with σ is shown in Fig. 22. For the multimodal spherical data, it is believed that these results closely approximate the optimum performance, since the overlap between subclasses was not great, and since subclasses of the same category turned out to be very rarely confused.

The optimal classifier for the unimodal spherical data is merely a minimum-distance classifier²; a pattern is classified by measuring its distance from the average (mean) of the patterns in each category, and by assigning it to the category corresponding to the nearest mean. While such a classifier is a reasonable choice for unimodal data, it is not well suited for multimodal data. When such a classifier was used with the multimodal spherical patterns, it misclassified 40 percent of the patterns with $\sigma = 0$, and 54 percent with $\sigma = 25$.

TABLE I SUBCLASS ASSIGNMENTS FOR MULTIMODAL DATA

Unimodal Category	Multimodal Category	Unimodal Category	Multimodal Category
1	1	25, 26, 27	17
2	2	28, 29, 30	18
3	3	31, 32, 33	19
4	4	34, 35, 36	20
5	5	37, 38, 39	21
6	6	40, 41, 42	22
7	7	43, 44, 45	23
8	8	46, 47, 48	24
9, 10	9	49, 50, 51, 52	25
11, 12	10	53, 54, 55, 56	26
13, 14	11	57, 58, 59, 60	27
15, 16	12	61, 62, 63, 64	28
17, 18	13	65, 66, 67, 68	29
19, 20	14	69, 70, 71, 72	30
21, 22	15	73, 74, 75, 76	31
23, 24	16	77, 78, 79, 80	32

The ellipsoidal data were obtained by subjecting the noise vectors for the spherical data to linear transformations. Different transformation matrices were used for each mode, so that while the surfaces of equal probability density were always hyperellipsoids centered about the prototype vectors, these hyperellipsoids had different shapes and orientations. The stretching of the hyperspheres and the rotation of the resulting hyper-

ellipsoids were done randomly, under the constraints that the volumes remained unchanged and that the ratios of the lengths of principal axes never exceeded 10:1. Although an optimal classifier for the unimodal ellipsoidal data is known,² the computation of the optimum performance is difficult and was not attempted. Instead, a minimum-distance classifier was used to give a relative performance standard. The variation in its performance with σ is shown in Fig. 22. As with the spherical data, this performance can also be regarded as a standard for the multimodal ellipsoidal data.

One interesting interpretation of the problem of classifying these patterns is to consider it as a problem in signal identification. The ten components of each of the 80 prototype vectors can be viewed as ten time samples of signals that are to be transmitted over a noisy channel. The components of the noisy vectors correspond to time samples of the received signal. The classification problem is to determine the category of the transmitted signal from the ten samples of the received signal.

Some feeling for the difficulty of this problem can be gained by examining the 80 prototype signals shown in Fig. 23. These have been sorted into 32 categories according to the assignments given by Table I. Note in particular the similarity between the first signal in Category 18 and the second signal in Category 19; the addition of any appreciable amount of noise makes the separation of these two categories particularly difficult.

C. THE ERROR-CORRECTION RULE USED WITH THE LINEAR AND PWL MACHINES

For both the linear and PWL machines, the error-correction rule calls for the addition of the pattern vector to one of the weight vectors and the

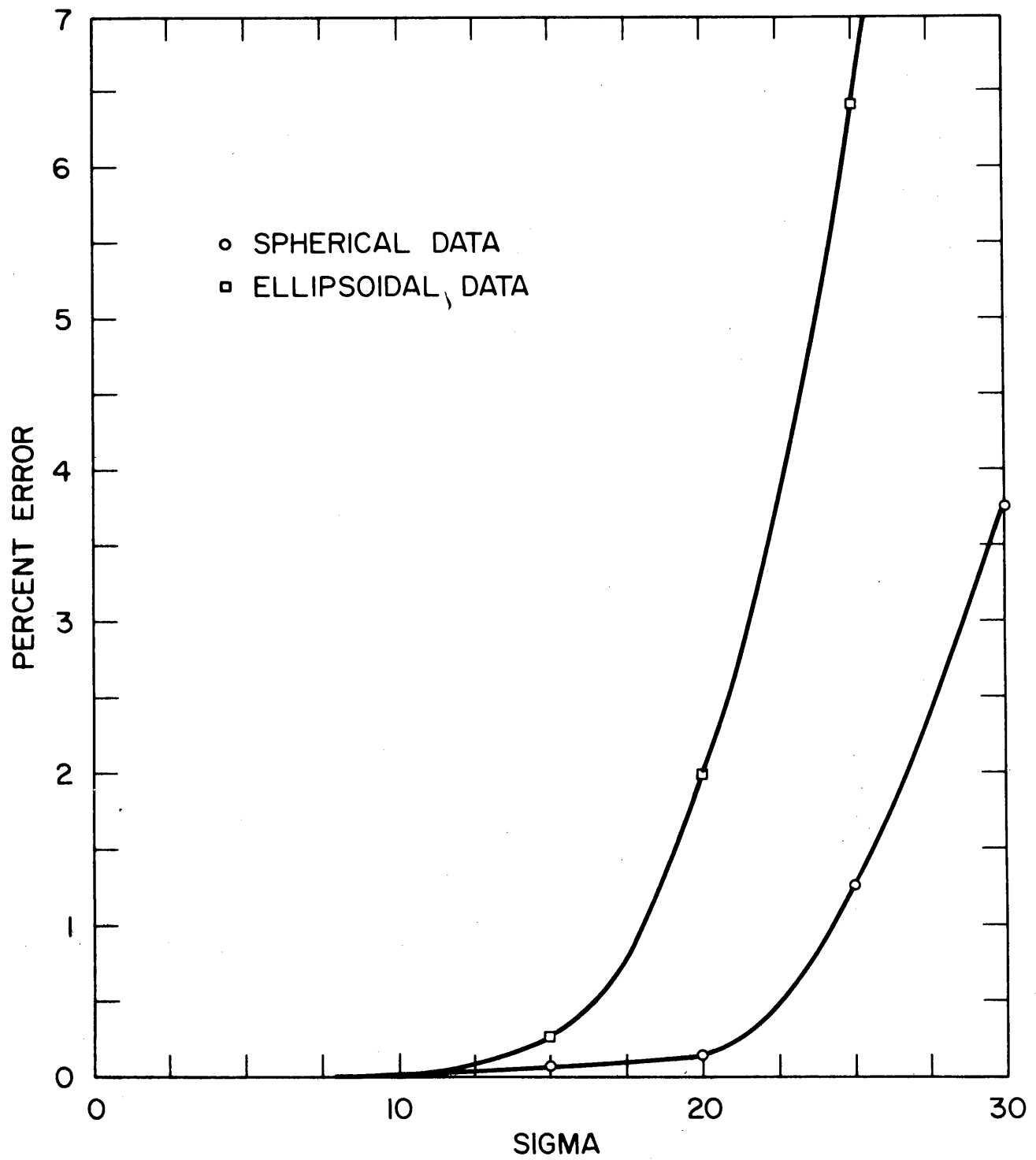


FIG. 22 PERFORMANCE OF A MINIMUM-DISTANCE CLASSIFIER
ON UNIMODAL GAUSSIAN DATA

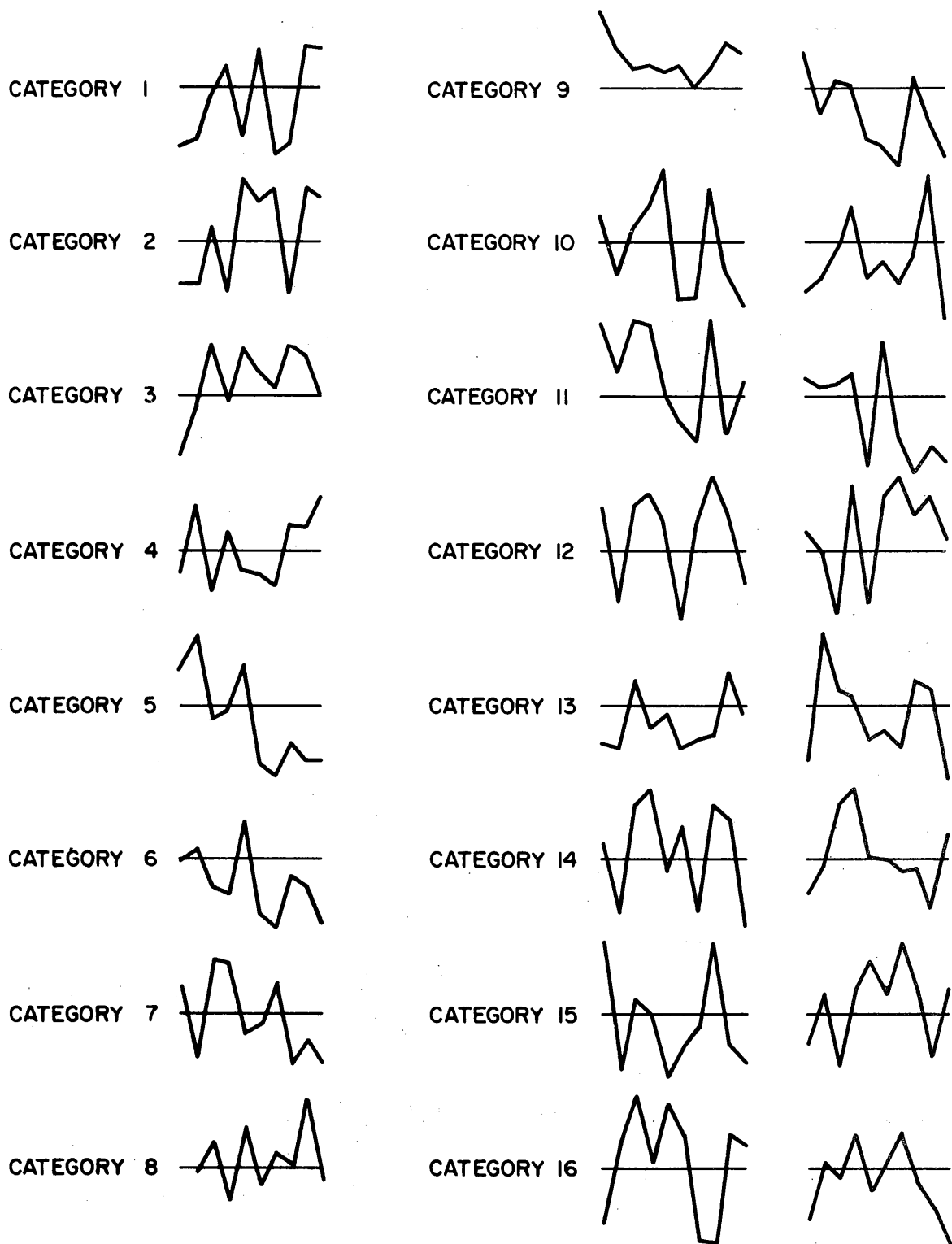


FIG. 23 WAVEFORM REPRESENTATION OF PROTOTYPE PATTERN VECTORS

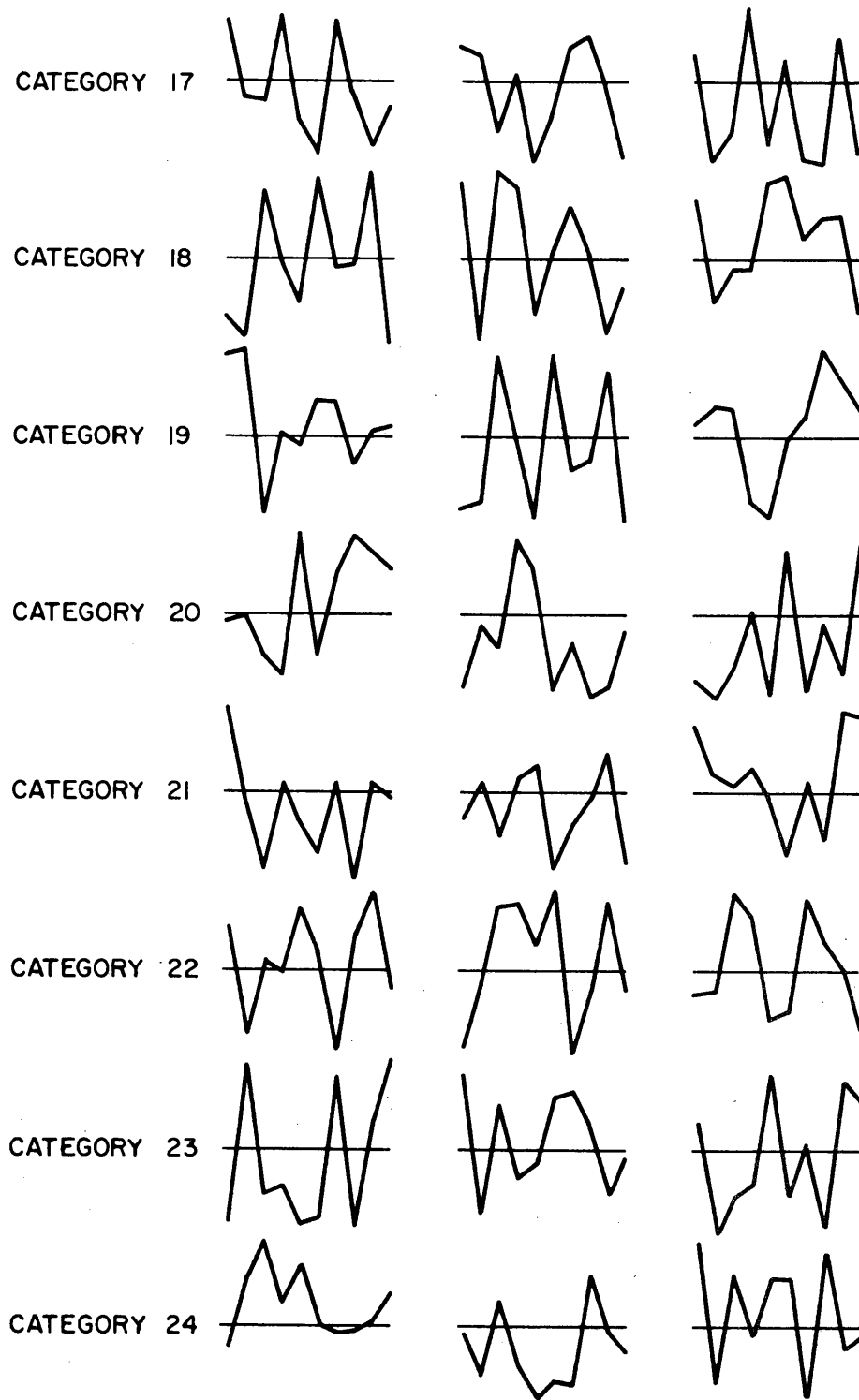


FIG. 23 Continued

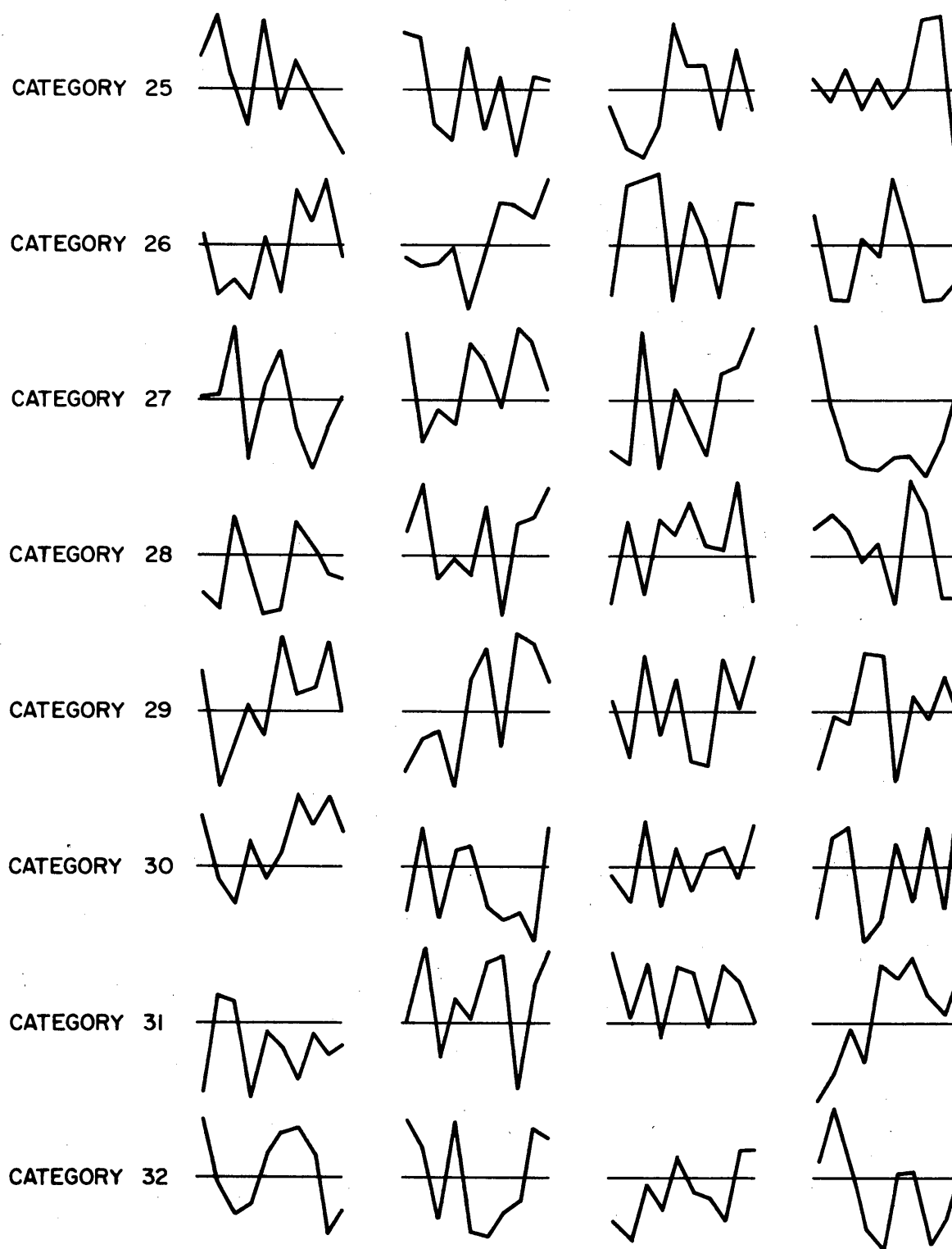


FIG. 23 Concluded

subtraction of the pattern vector from another of the weight vectors whenever a pattern vector is classified incorrectly. During the experiments to be described, two elaborations were added to this basic rule. First, instead of adding or subtracting the pattern vector itself, a constant a_k times the pattern vector was added or subtracted. The correction increment, a_k , was allowed to vary with the number of corrections, k , but must satisfy: $0 < a_{\min} \leq a_k \leq a_{\max}$. Second, instead of making a correction only for erroneously classified patterns, a correction was also made if the largest dot product failed to exceed the dot products in the other banks by at least a margin, M .

For linear machines, the above rule is a simple modification of the error-correction rule first proposed by Kesler.¹¹ Both are guaranteed to result in error-free solutions provided only that such a solution exists. For the PWL machine, our knowledge of the convergence properties of the above rule is mainly empirical. It is known that the mere existence of a solution, i.e., the existence of a PWL machine that correctly classifies all of the training patterns, is not sufficient to guarantee convergence. Cyclic presentation of patterns, for example, can lead to cyclic variations in the weights.

Even if this rule were guaranteed to achieve a solution for a PWL machine whenever a solution exists, there would remain the problem that few pattern-recognition problems have error-free solutions. Thus, to be useful, the correction procedure must be well behaved even when convergence is not possible. The parameters M and a_k are important in this regard. By selecting a sufficiently large margin M relative to the average value of

$a_k |\vec{x}|^2$, the disturbances created by correcting for occasional highly noisy patterns can be reduced. Reducing the increment factor a_k during training also has a stabilizing effect, and it has been found to be useful in accelerating convergence. In the experiments to be described, a_k was changed only after the presentation of a block of N patterns. If \bar{a}_n was its value during the n th block, and f_n was the fraction of patterns in that block requiring correction, the value for the next block was determined from

$$\begin{aligned}\bar{a}_1 &= a_{\max} \\ \bar{a}_{n+1} &= \max \left[a_{\min}, \frac{1}{2}(\bar{a}_n + a_{\max} f_n) \right] \\ n &= 1, 2, \dots\end{aligned}\tag{23}$$

Thus, the increment factors were decreased as the short-term correction rate decreased, but they were never allowed to fall below a_{\min} . The actual selection of the parameters M, N, a_{\min} , and a_{\max} was done experimentally.

D. EXPERIMENTS USING THE TEN-DIMENSIONAL GAUSSIAN PATTERNS

The primary purpose of the experimental study was to answer the following questions:

- (1) How long does it take the training process to converge?
- (2) What is the limiting error rate for the training data?
For independent testing data?
- (3) How does the performance depend upon the following characteristics of the classifier?
 - (a) Linear or piecewise linear structure
 - (b) Margin, M

- (c) Increment factor parameters: N , a_{\min} ,
 a_{\max} .

- (4) How does the performance depend upon the following characteristics of the data?
- (a) Unimodal or multimodal
 - (b) Spherical or ellipsoidal
 - (c) Standard deviation of noise, σ
 - (d) Number of training patterns.

We shall present experimental results that supply at least partial answers to these questions. We begin by describing a typical experiment in which a linear machine, implemented on an SDS 910 computer, was trained on unimodal spherical data with $\sigma = 25$. A total of 3200 sample patterns was generated, each mode being represented by 40 noisy samples. Half of the patterns were used to train the machine, and the remaining half were used to provide an independent test. The inputs to the linear machine consisted of the ten components of the pattern vector plus an eleventh component arbitrarily fixed at the value 30. A margin of 10,000 was used ($M/|\vec{x}|^2 \cong 1/3$), and the increment factor parameters were $N = 400$, $a_{\min} = 1$, and $a_{\max} = 3$.

The training patterns were presented to the machine cyclically, each pass through the patterns being termed a cycle. During each cycle the weights were held fixed and the number of errors made on the independent testing data was recorded. For these error counts the margin was always ignored; a classification of a pattern in Category i was considered to be correct if the i th dot product was larger than any of the others.

The results of this run are shown in Fig. 24. For this problem, the error rate of an optimal classifier is 1.25 percent (see Fig. 22), and after six iterations the running error rate for the training data had fallen below this value. At about the same time, the performance on the independent testing data reached and began fluctuating about its limiting value of approximately 4 percent.

These results are qualitatively characteristic of the results of most of the experiments performed. When the variance of the noise was small and the training patterns were linearly separable, true convergence was achieved. More typically the training patterns were not linearly separable; in these cases, the error rate for the training data dropped rapidly at first and eventually fluctuated about some limiting value. The error rate for the testing data stabilized at a higher limiting value considerably sooner, the difference between the performance on training and testing data decreasing as the number of training patterns was increased.

The performance was not found to be particularly sensitive to variations in the machine parameters. The simplest choices are $M = 0$, $N = \infty$, and $a_{\min} = a_{\max} = 1$. Definite improvement in performance resulted by choosing M near $a_{\min} \left| \tilde{x} \right|^2$, N several times the number of modes, and a_{\max} several times a_{\min} . None of these choices appeared to be critical, however, and an exhaustive investigation was not attempted.

The tree shown in Fig. 25 summarizes the results obtained when the characteristics of the data were varied. Except for the variations indicated by the branches of this tree, the experiments were performed under the conditions described previously. The principal results, viz., the minimum percentage error for the independent testing data, are listed at the nodes of the tree. A few comments about these results are in order.

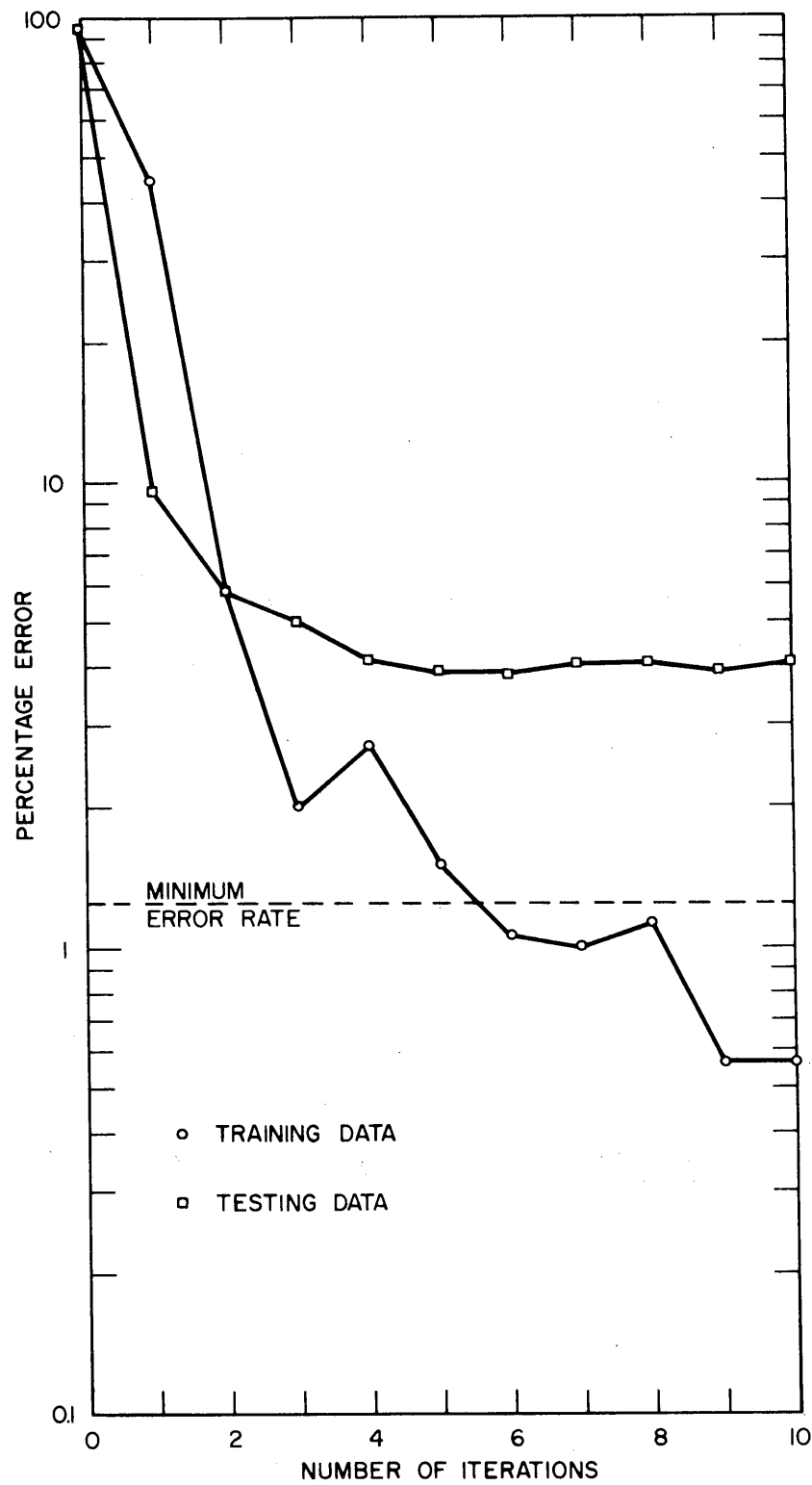


FIG. 24 PERFORMANCE OF A LINEAR MACHINE TRAINED ON UNIMODAL SPHERICAL DATA

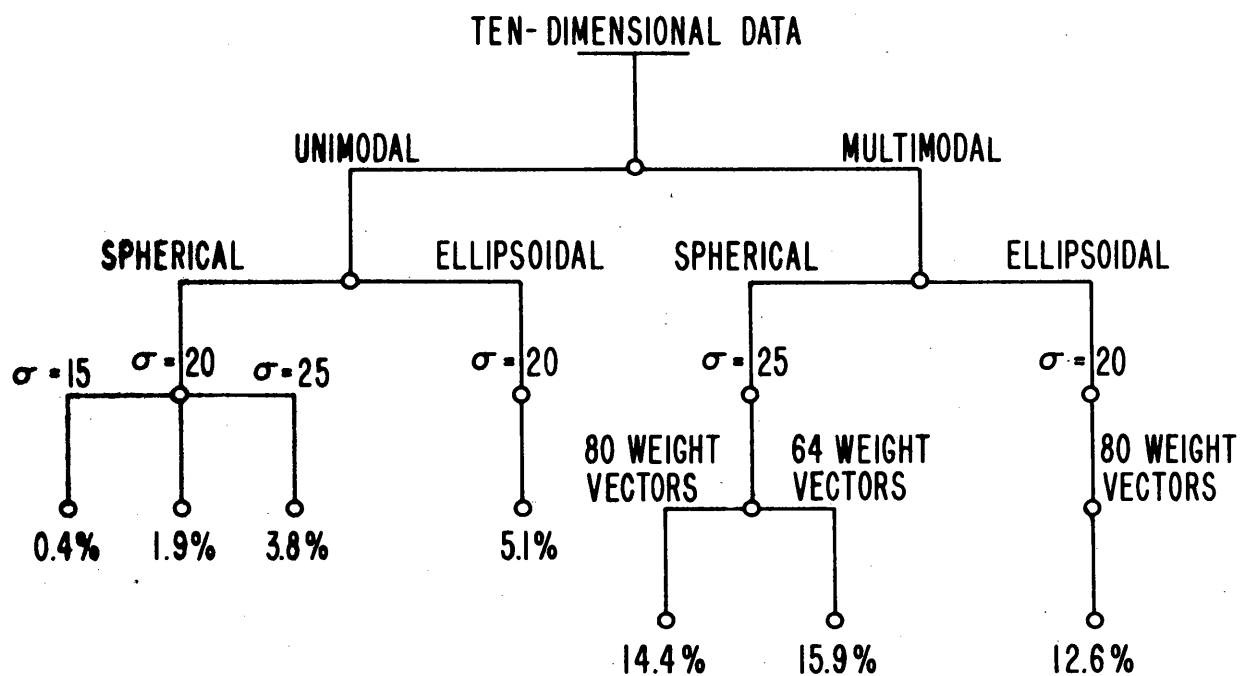


FIG. 25 EXPERIMENTAL RESULTS (PERCENTAGE ERROR ON TESTING PATTERNS) FOR GAUSSIAN DATA

The performance obtained with the unimodal spherical data for a given value of σ was roughly the same as the optimum performance with σ replaced by $\sigma + 5$ (compare Figs. 22 and 25). Thus the percentage discrepancy was relatively high when the optimum error rate was low. Only one run was made with the unimodal ellipsoidal data, and the error rate achieved was about 2-1/2 times the error rate of a minimum-distance classifier. Undoubtedly, these results could be further improved by increasing the number of training patterns.

A piecewise linear machine was used in the experiments with the multimodal data. In some experiments the machine structure was perfectly matched to that of the data; 80 weight vectors were used, each of the 32 banks containing the same number of weight vectors as there were modes in the corresponding category. Because the detailed structure of the data would presumably not be known in practical applications, in the other experiments 64 weight vectors were used, two weight vectors being arbitrarily assigned to each bank. In both cases the error rates achieved were very similar; moreover, they were noticeably higher than the corresponding error rates for unimodal data, being about 4 times higher for spherical data and 2-1/2 times higher for ellipsoidal data.

The similarity in the performance obtained with 64 and 80 weight vectors suggests that, in the latter case at least, all of the available weight vectors were not being used. Examination of the weights showed this to be the case in both instances. Part of this can be attributed to the fact that it is not necessary for a piecewise linear machine to have a weight vector for each mode. However, part must be attributed to the

simplicity of the rule used for assigning patterns to subclasses, viz., to select the subclass corresponding to the largest dot product in the correct bank. This rule may be inadequate when pattern vectors that should be placed in different subclasses are close to one another, and appears to become progressively less effective as the number of weight vectors per bank is increased. It appears that more than three weight vectors per bank cannot be effectively trained using the present version of the rule. Despite these limitations, the simplicity of the rule makes it attractive, particularly if the increased error can be tolerated to avoid complex data analysis.

E. EXPERIMENTS WITH 100-DIMENSIONAL, QUANTIZED GAUSSIAN PATTERNS

During these experiments the question arose as to how quantization of the input variables would affect performance. This question arose because the classifier part of MINOS II, an experimental learning machine built by Stanford Research Institute for the U.S. Army Electronics Laboratories, accepts binary (+1, -1) rather than analog inputs.^{6,16} Use of this classifier with the Gaussian data would require that the 10 analog variables x_i be quantized and represented as 100 binary variables. It was decided to do this by dividing the noise-free range of each variable (-99 to 99) into ten equal intervals I_j , the end intervals being then extended to infinity. The 100 binary inputs were determined in groups of ten as follows: if the ith analog variable fell in the jth quantization interval, the first j inputs in the ith group were +1's, and the remaining (10 - j) inputs were -1's. Two additional constant (+1) inputs were included as the so-called threshold inputs.

The multimodal data used in the previous experiments were quantized and coded in this way, and were used to train a simulated MINOS classifier, in the form of a PWL machine having 102 binary inputs and 64 weight vectors, two weight vectors per bank. The machine parameters were $m = 150$, $N = 400$, $a_{\min} = 1$, and $a_{\max} = 3$. The results obtained were very similar to those for the analog data. The error rate on the multimodal spherical data with $\sigma = 25$ and 1600 training patterns was 11.9 percent which was actually lower than the 15.9 percent for the analog data. However, for the ellipsoidal data the error rate was 14.6 percent compared to 12.6 percent for the analog data.

A final set of experiments was performed to determine the effect of the number of training patterns on performance; the results are shown in Fig. 26. As expected, increasing the number of training patterns both improved the performance on independent testing data and reduced the discrepancy between the performance on training and testing data. The price for this improvement was increased computation time. Our simulations on the SDS 910 of a machine containing 64 weight vectors ran at about 4.5 patterns/sec, so that ten iterations of 1600 patterns required an hour. As can be seen from Fig. 26, computing time would have to be increased considerably to gain a modest increase in performance.

F. EXPERIMENTS WITH COMMITTEE MACHINES

One other experiment was performed using the 32 category, multimodal, 100-dimensional, quantized Gaussian data. In this experiment, a network of committee machines implemented on the SDS-910 computer was trained on 1600 patterns using a noise standard deviation of $\sigma = 15$.

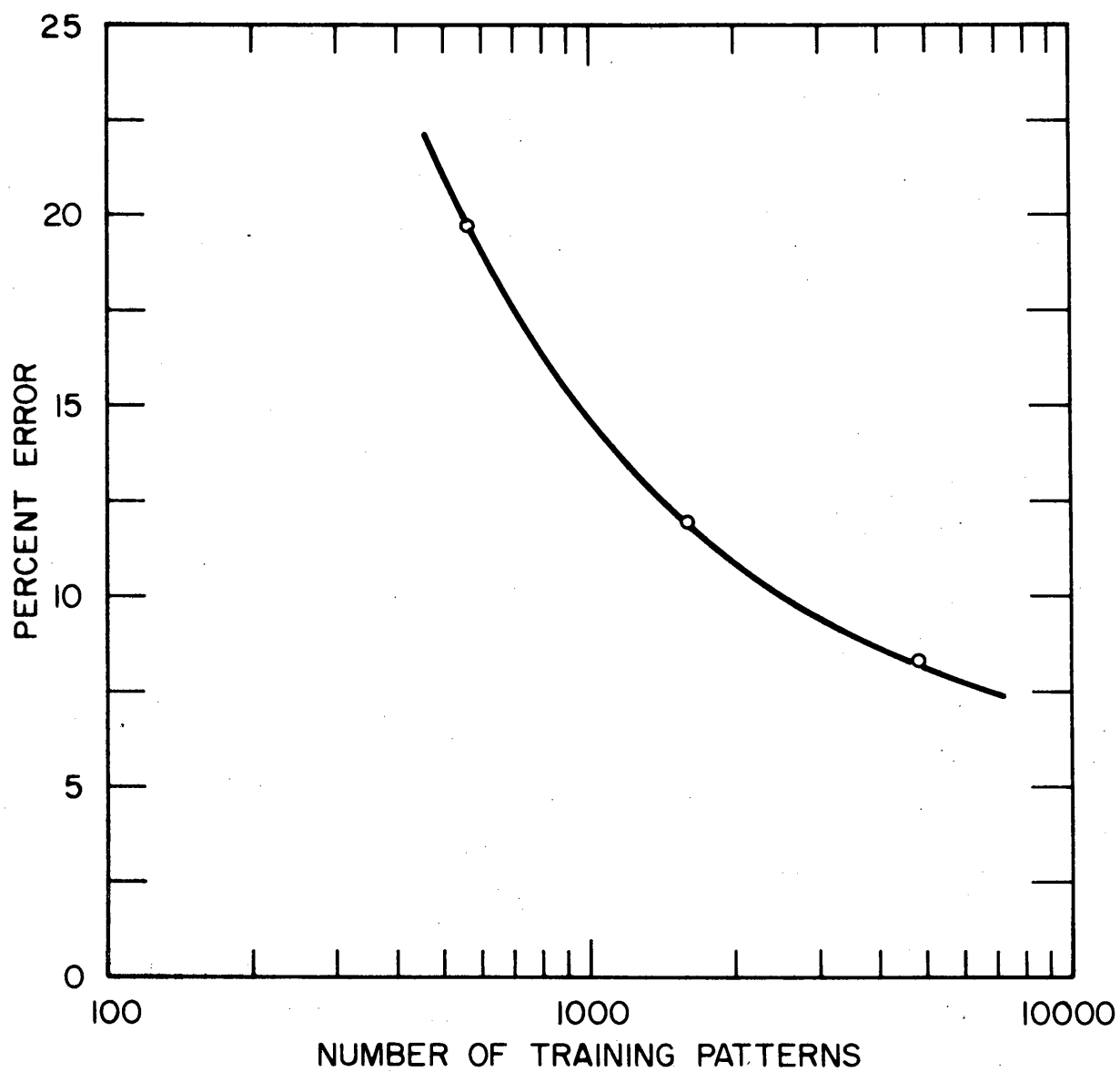


FIG. 26 EFFECT OF SAMPLE SIZE ON PERFORMANCE

The network was organized to give a 32 category response as follows: nine committee machines, each composed of seven TLU's, comprised a parallel organization of nine dichotomizers as discussed in Sec. I-D-2. An error-correcting code was used with the nine-bit output to correct any single errors in the 32 legal output code words. Thus the whole network had a total of $7 \times 9 = 63$ weight vectors and was comparable in size with the PWL machine used on the same quantized Gaussian data.

Each committee machine was trained according to the standard committee error-correction rule proposed by Ridgway¹⁷ and used in conjunction with MINOS II.⁶ The error-correcting properties of the output code were ignored during training, and each committee machine was trained whenever its binary output disagreed with the desired output.

During testing on independent data, the error-correcting properties of the output code were employed; thus, single errors that occurred were not counted as classification errors since they would have been corrected.

The training curve for this experiment is shown in Fig. 27. The ordinate, "percent errors," is the percentage of the 1600 training patterns causing two or more committee errors during a cycle.

In this experiment, the network of committee machines was trained down to zero errors on the 1600 training patterns after 33 cycles. When tested on 1600 independent patterns, it made 190 classification errors, however. These 190 errors represented about a 12 percent error rate, which was achieved by the PWL machine operating on much noisier data ($\sigma = 25$).

Exhaustive comparisons between the committee machines organization and the PWL machine have not been conducted, however, and it would be premature

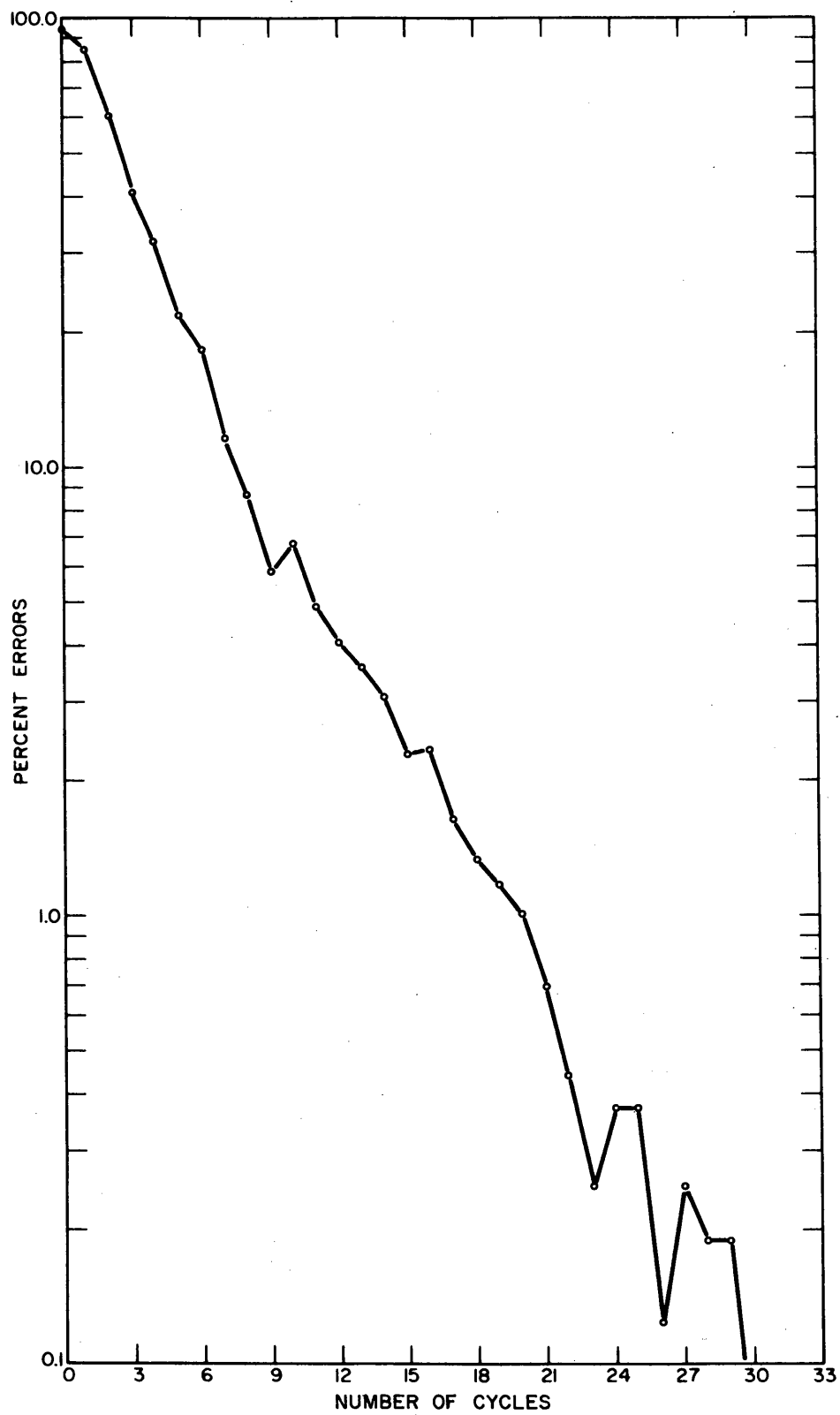


FIG. 27 PERFORMANCE OF COMMITTEE MACHINES TRAINED ON MULTIMODAL, QUANTIZED GAUSSIAN DATA

to infer that the PWL machine is the superior organization. It is hoped that future experiments can be designed and executed that will illuminate any differences in performance between these two classifiers for various kinds of input patterns.

IV SUMMARY AND CONCLUSIONS

In this report we have attempted to present arguments to make plausible the use of certain classifier structures called linear and PWL machines. The study of these structures and some others (e.g., committee machines) has been at the center of research conducted during the present project. There are several types of questions that should be answered by such a study. First, we are interested in the subject of training these structures. The subject of training has received the major attention during the present project. Second, we are interested in other properties of these structures such as their pattern-classifying capacity and the nature of the decision rules implemented by them. Some research on these properties has been conducted and reported in two separate technical documentary reports entitled: "A Committee Solution of the Pattern Recognition Problem," by Ablow and Kaylor, and "Notes on Classification Capacities," by Cover.

We have used both theoretical and experimental avenues in conducting research on the training problem. First there has been some theoretical work on the subject of training linear machines. The results of this work are presented in an Appendix to this report. Theoretical attacks on other items--such as training PWL machines and training committee machines--have been largely unsuccessful, and our knowledge here has been obtained through experiments.

In this report we have presented the results of several training experiments that investigated various methods of training PWL machines. Our explanation of these methods was illustrated with various two-dimensional experiments, and we later described more thorough higher dimensional experiments. Two main types of training methods were used: Mode-seeking and error-correction methods. Of the mode-seeking methods, the best results were obtained with the Rosen-Hall method. The error-correction methods often give quite good results. In some respects they are simpler to apply, but further research is needed to alleviate some of their attendant difficulties.

The state of knowledge about training these pattern-classifying structures can be summarized as follows: Several plausible training methods have been suggested. The PWL machine error-correction training method has now been extensively investigated, and it has been found to be quite effective in a variety of pattern-classifying experiments. As it stands now, however, it is probably not a very effective method for training PWL machines having more than three DPU's in each bank. Research on the mode-seeking training methods has been sufficient to expect that they will be quite useful for some problems but extensive experience with these methods has yet to be obtained.

Several topics are in need of further exploration and we shall conclude by listing some of them.

To expand our knowledge of the properties of classifying structures research should be conducted on the following items:

- (1) Pattern-classifying capacity of classifying structures.
- (2) Use of various output codes in parallel organizations of dichotomizers.

- (3) Experimental comparison of performance of various classifying structures on some sample problems.

To expand our knowledge of training methods, research should be conducted on the following items:

- (1) Further experiments using Rosen-Hall method.
- (2) Investigation of weight vector birth and death rules to be used in error-correction training of PWL machines.
- (3) Further theoretical research aimed at establishing training algorithms with guaranteed convergence properties.

In addition to the above items, concurrent efforts should be made to identify those real-world pattern classifying problems to which this developing body of techniques can be usefully applied.

APPENDIX I

by R. O. Duda

This appendix contains convergence proofs for some error-correction training rules for linear machines. These proofs are basically generalizations of Novikoff's proof of the Perceptron Convergence Theorem.¹⁸ In all cases we assume that we are given a fixed, finite, linearly separable set of training patterns. That is, we assume that there exists a linear machine, called a solution machine, that correctly classifies all of the training patterns. Equivalently, we assume that there exists a set of solution weight vectors $\{\hat{\underline{W}}_i\}$ ($i = 1, \dots, R$) such that for any category ω_i and any pattern vector \underline{X} in ω_i

$$\hat{\underline{W}}_i^t \underline{X} > \hat{\underline{W}}_j^t \underline{X} \quad (1)$$

for all $j \neq i$, ($i, j = 1, \dots, R$).*

These patterns, finite in number, are presented to the linear machine in a training sequence, any infinite sequence in which each pattern recurs infinitely often. Let \underline{X}_k denote the k th pattern in the training sequence leading to a correction, let $\{\underline{W}_i(k)\}$ ($i = 1, \dots, R$) denote the set of weight vectors just preceding the k th correction, and let $\{\underline{C}_i(k)\}$ ($i = 1, \dots, R$) denote the k th set of correction vectors. The set of initial weight vectors $\{\underline{W}_i(1)\}$ is arbitrary, and the subsequent weight vectors are found from

$$\begin{aligned} \underline{W}_i(k+1) &= \underline{W}_i(k) + \underline{C}_i(k) & i = 1, \dots, R \\ & & k = 1, 2, \dots \end{aligned} \quad (2)$$

* There are some notational differences between this appendix and the main body of the report. In the appendix, vectors are denoted by a wavy underline (e.g. \underline{X}). To apply the results of this appendix to the linear machines discussed in the report the reader should assume that the pattern and weight vectors (\underline{X} and \underline{W}) of the appendix are $(d+1)$ -dimensional. The $(d+1)$ -th component of a pattern is fixed at a value of $+1$ so that the dot product $\underline{X} \cdot \underline{W}$ contains the bias term W_{d+1} .

The correction vectors $\underline{C}_i(k)$ are related to the pattern vectors \underline{X}_k by the training rule. Before we assume any particular relation, we shall establish conditions on the correction vectors that are sufficient to guarantee the termination of the correction process. To simplify notation, we shall write \underline{W}_i and \underline{C}_i instead of $\underline{W}_i(k)$ and $\underline{C}_i(k)$ when there is no danger of confusion; however, it should be kept in mind that these conditions must be satisfied for all k .

Lemma: Let $\underline{W}_i(k+1) = \underline{W}_i(k) + \underline{C}_i(k)$ ($i = 1, \dots, R; k = 1, 2, \dots$) with $\{\underline{W}_i(1)\}$ arbitrary. Then k is bounded if there exist constants a , b , and c , and a set of constant vectors $\{\hat{\underline{W}}_i\}$ ($i = 1, \dots, R$) such that for any k

$$(A): \quad \sum_{i=1}^R \sum_{j=1}^R (\underline{C}_i - \underline{C}_j)^t (\underline{C}_i - \underline{C}_j) \leq a, \quad ,$$

$$(B): \quad \sum_{i=1}^R \sum_{j=1}^R (\underline{W}_i - \underline{W}_j)^t (\underline{C}_i - \underline{C}_j) \leq b, \quad ,$$

and

$$(C): \quad \sum_{i=1}^R \sum_{j=1}^R (\hat{\underline{W}}_i - \hat{\underline{W}}_j)^t (\underline{C}_i - \underline{C}_j) \geq c > 0 \quad .$$

The proof is obtained by deriving conflicting bounds on the growth of the quantity S_k defined by

$$S_k^2 = \sum_{i=1}^R \sum_{j=1}^R \left[\underline{W}_i(k) - \underline{W}_j(k) \right]^t \left[\underline{W}_i(k) - \underline{W}_j(k) \right] \quad . \quad (3)$$

To obtain an upper bound, we note that from Eqs. (2) and (3)

$$\begin{aligned}
 s_{k+1}^2 = & s_k^2 + 2 \sum_{i=1}^R \sum_{j=1}^R (\underline{w}_i - \underline{w}_j)^t (\underline{c}_i - \underline{c}_j) \\
 & + \sum_{i=1}^R \sum_{j=1}^R (\underline{c}_i - \underline{c}_j)^t (\underline{c}_i - \underline{c}_j)
 \end{aligned} \quad (4)$$

so that by Conditions (A) and (B)

$$s_{k+1}^2 \leq s_k^2 + 2b + a \quad . \quad (5)$$

It follows that

$$s_{k+1} \leq \sqrt{k(2b + a) + s_1^2} \quad (6)$$

where the value of s_1^2 depends only on the choice of the initial weight vectors.

To obtain a lower bound on S_k , consider the quantity T_k defined by

$$T_k = \sum_{i=1}^R \sum_{j=1}^R [\hat{\underline{w}}_i - \hat{\underline{w}}_j]^t [\underline{w}_i(k) - \underline{w}_j(k)] \quad . \quad (7)$$

By Eq. (2) and Condition (C),

$$T_{k+1} = T_k + \sum_{i=1}^R \sum_{j=1}^R (\hat{\underline{w}}_i - \hat{\underline{w}}_j)^t (\underline{c}_i - \underline{c}_j) \quad (8)$$

$$\geq T_k + c \quad . \quad (9)$$

It follows that

$$T_{k+1} \geq k c + T_1 \quad (10)$$

where the value of T_1 depends on the initial weight vectors, but is independent of k . Now by the Cauchy-Schwarz inequality,

$$\sqrt{\sum_{i=1}^N A_i^t A_i} \sqrt{\sum_{i=1}^N B_i^t B_i} \geq \sum_{i=1}^N A_i^t B_i \quad (11)$$

Applying this to Eq. (7), we obtain

$$|\hat{\tilde{W}}| S_k \geq T_k \quad (12)$$

where

$$|\hat{\tilde{W}}| = \sqrt{\sum_{i=1}^R \sum_{j=1}^R (\hat{\tilde{W}}_i - \hat{\tilde{W}}_j)^t (\hat{\tilde{W}}_i - \hat{\tilde{W}}_j)} \quad (13)$$

Combining Eqs. (10) and (12), we obtain the desired lower bound:

$$S_{k+1} \geq k \frac{c}{|\hat{\tilde{W}}|} + \frac{T_1}{|\hat{\tilde{W}}|} \quad (14)$$

Since $c/|\hat{\tilde{W}}|$ is strictly positive, if k were not bounded we could choose k so large that the lower bound, Eq. (14), would exceed the upper bound, Eq. (6). It follows that k must be bounded.

We shall now use this result to prove that our generalization of Kesler's training rule yields a solution machine after a finite number

of corrections. If a pattern \underline{x}_k belonging to ω_m is presented, the rule calls for a correction if and only if $\underline{w}_m^t \underline{x}_k - M \leq \underline{w}_n^t \underline{x}_k$ for some $n \neq m$, where M is the non-negative margin. By letting \underline{x}_k denote the k th pattern in the training sequence leading to a correction, we can write for all k

$$\underline{w}_m^t(k) \underline{x}_k - M \leq \underline{w}_n^t(k) \underline{x}_k \quad (15)$$

for n in some nonempty set of integers N_k , $m \notin N_k$. Then the training rule can be written in terms of the correction vectors as follows:

$$\underline{c}_i(k) = \begin{cases} a_k \underline{x}_k & i = m \\ -\alpha_{nk} a_k \underline{x}_k & i = n, n \in N_k \\ 0 & \text{otherwise} \end{cases}, \quad (16)$$

where a_k and α_{nk} must satisfy

$$0 < a_{\min} \leq a_k \leq a_{\max}, \quad (17)$$

$$\alpha_{nk} \geq 0, \quad n \in N_k, \quad (18)$$

and

$$\sum_{n \in N_k} \alpha_{nk} = 1 \quad (19)$$

for all k . We shall call this rule the bounded-increment error correction rule. Starting with an initial linear machine, determined by a set of initial weight vectors $\{\tilde{w}_i(1)\}$ ($i = 1, \dots, R$), application of this rule to the vectors in a training sequence yields a sequence of linear machines. We shall now demonstrate the convergence of this sequence.

Theorem: Let \mathcal{L}_1 be any initial linear machine, and let $\{\mathcal{L}_i\}$ ($i = 1, 2, \dots$) be the sequence of linear machines resulting from the application of the bounded-increment error correction rule to any training sequence of linearly separable patterns. Then the sequence $\{\mathcal{L}_i\}$ will terminate in a solution linear machine after a finite number of corrections.

The proof is obtained by showing that the correction vectors for this training rule satisfy the conditions of the lemma. Noting from Eqs. (16) and (19) that

$$\sum_{i=1}^R \tilde{c}_i(k) = 0 \quad (20)$$

and using Eqs. (16), (17), (18), and (19), we obtain

$$\begin{aligned} \sum_{i=1}^R \sum_{j=1}^R (\tilde{c}_i - \tilde{c}_j)^t (\tilde{c}_i - \tilde{c}_j) &= 2R \sum_{i=1}^R \tilde{c}_i^t \tilde{c}_i - 2 \left[\sum_{i=1}^R \tilde{c}_i \right]^t \left[\sum_{i=1}^R \tilde{c}_i \right] \\ &= 2R a_k^2 \sum_{\tilde{k}} \tilde{x}_k^t \tilde{x}_k (1 + \sum_{n \in N_k} \alpha_{nk}^2) \\ &\leq 4R a_{\max}^2 \sum_{\tilde{k}} \tilde{x}_k^t \tilde{x}_k \quad . \end{aligned} \quad (21)$$

Since the training patterns are fixed and finite in number, it follows that Condition (A) is satisfied. For Condition (B), we first obtain from Eqs. (16) and (20)

$$\begin{aligned} \sum_{i=1}^R \sum_{j=1}^R (\underline{w}_i - \underline{w}_j)^t (\underline{c}_i - \underline{c}_j) &= 2R \sum_{i=1}^R \underline{w}_i^t \underline{c}_i - 2 \left[\sum_{i=1}^R \underline{w}_i \right]^t \left[\sum_{i=1}^R \underline{c}_i \right] \\ &= 2R a_k \left[\underline{w}_m^t \underline{x}_k - \sum_{n \in N_k} \alpha_{nk} \underline{w}_n^t \underline{x}_k \right] \quad (22) \end{aligned}$$

It then follows readily from Eqs. (15), (17), (18), and (19) that

$$\sum_{i=1}^R \sum_{j=1}^R (\underline{w}_i - \underline{w}_j)^t (\underline{c}_i - \underline{c}_j) \leq 2R a_{\max} M \quad (23)$$

so that Condition (B) is also satisfied. For Condition (C), we first obtain from Eqs. (16) and (20) a similar expansion involving the solution weight vectors:

$$\sum_{i=1}^R \sum_{j=1}^R (\hat{\underline{w}}_i - \hat{\underline{w}}_j)^t (\underline{c}_i - \underline{c}_j) = 2R a_k \left[\hat{\underline{w}}_m^t \underline{x}_k - \sum_{n \in N_k} \alpha_{nk} \hat{\underline{w}}_n^t \underline{x}_k \right] \quad (24)$$

Since $\underline{x}_k \in \omega_m$ and $\{\hat{\underline{w}}_i\}$ is a set of solution vectors,

$$\hat{\underline{w}}_m^t \underline{x}_k \geq \hat{\underline{w}}_n^t \underline{x}_k + d \quad (25)$$

where

$$d = \min_i \left[\min_{j \neq i} \left\{ \min_{\underline{x} \in \omega_i} (\hat{\underline{w}}_i - \hat{\underline{w}}_j)^t \underline{x} \right\} \right] \quad (26)$$

Since the patterns are finite in number and linearly separable, d is strictly positive. It follows readily from Eqs. (24), (25), (17), (18), and (19) that

$$\sum_{i=1}^R \sum_{j=1}^R (\hat{w}_i - \hat{w}_j)^t (C_i - C_j) \geq 2R a_{\min} d \quad (27)$$

and hence that Condition (C) is satisfied. Thus, by the lemma, k is bounded and the number of corrections is finite. But since each training pattern recurs infinitely often in the training sequence, this implies that after the corrections cease $\hat{w}_m^t X - M > \hat{w}_n^t X$ for any category ω_m , for any training pattern X belonging to ω_m , and for all $n \neq m$. Since the margin M is non-negative, the terminal weight vectors are solution vectors, and thus the theorem is proved.

Generalizations

In conclusion, we present some generalizations and alternative rules without proofs. From a theoretical viewpoint, the conditions on the increment factor a_k given by Eq. (17) are quite restrictive. A more careful analysis allows this condition to be relaxed somewhat, replacing it by the two conditions

$$a_k > 0 \quad k = 1, 2, \dots \quad (28)$$

and

$$\lim_{n \rightarrow \infty} \left[\left(\sum_{k=1}^n a_k \right)^2 - \sum_{k=1}^n a_k^2 \right] = +\infty \quad (29)$$

The choices $a_k = 1/k$ or $a_k = k$ meet these conditions, for example, whereas $a_k = 1/k^2$ or $a_k = e^k$ do not. Other more interesting rules also fail to meet these conditions. One such rule takes a_k to be the smallest integer sufficient to correct the response. To be more specific, if $\tilde{x}_k \in \mathcal{M}_m$ and $\tilde{w}_n^t(k) \tilde{x}_k$ is the largest dot product, the rule is

$$\begin{aligned}
 a_k \tilde{x}_k & \quad i = m \\
 C_i(k) = & \quad -a_k \tilde{x}_k \quad i = n \\
 & \quad 0 \quad \text{otherwise}
 \end{aligned} \tag{30}$$

where a_k is the smallest integer greater than $(\tilde{w}_n^t \tilde{x}_k - \tilde{w}_m^t \tilde{x}_k + M) / 2\tilde{x}_k^t \tilde{x}_k$.

It can be shown that this rule also yields a solution after a finite number of corrections; the proof parallels the proof given by Nilsson for the absolute correction rule for training a threshold logic unit.¹¹ Finally, there is a generalization of the so-called fractional correction rule¹¹ of Agmon¹⁹ and Motzkin and Schoenberg²⁰ in which

$$a_k = \lambda \frac{\tilde{w}_n^t \tilde{x}_k - \tilde{w}_m^t \tilde{x}_k + M}{2\tilde{x}_k^t \tilde{x}_k} \tag{31}$$

where $0 < \lambda \leq 2$. The properties of this rule are more complex, since corrections may or may not terminate. It can be shown that if the corrections terminate, a set of solution weight vectors is obtained, and that termination always occurs if $\lambda = 2$. Otherwise the sequence

$\{\tilde{w}_i(1)\}, \{\tilde{w}_i(2)\}, \dots$ converges to a limit $\{\tilde{w}_i(\infty)\}$ for which

$$\tilde{w}_m^t \tilde{X} - M \geq \tilde{w}_n^t \tilde{X} \quad (32)$$

for any $\tilde{X} \in \omega_m$ and for all (m, n) , $m \neq n$. If $M > 0$, this limit is a set of solution vectors.

REFERENCES

1. Nilsson, N.J., Learning Machines: Foundations of Trainable Pattern Classifying Systems, (McGraw-Hill Book Company, New York, N.Y., 1965).
2. Nilsson, op. cit., Chapt. 3.
3. Fix, E., and J. L. Hodges, Jr., "Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties," Report 4, Contract AF 41(128) - 31, Project 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas (February, 1951).
4. Widrow, B., "Generalization and Information Storage in Networks of Adaline 'Neurons'," in Yovits, Jacob, and Goldstein (eds.), Self-Organizing Systems--1962, pp. 435-461, (Spartan Books, Washington, D.C., 1962).
5. Kaylor, D. J., "A Mathematical Model of a Two-Layer Network of Threshold Elements," Rome Air Development Center Technical Documentary Report RADC-TDR-63-534 (March 1964).
6. Brain, A. E., et al., "A Large, Self-Contained Learning Machine," 1963 WESCON Paper 6.1 (August 1963).
7. Brain, A. E., et al., "Graphical Data Processing Research Study and Experimental Investigation", Report No. 8 (pp. 9-13) and No. 9 (pp. 3-10), Contract DA 36-039 SC-78343, Stanford Research Institute, Menlo Park, California (June 1962 and September 1962).
8. Peterson, W. W., Error-Correcting Codes, pp. 147 ff. (M.I.T. Press, Cambridge, Massachusetts, and John Wiley & Sons, Inc., New York, N.Y. 1961).
9. Nilsson, op. cit., Chapt. 2.
10. Nilsson, op. cit., Chapt. 6.
11. Nilsson, op. cit., Chapt. 4 and 5.
12. Stark, L., M. Okajima, and G. Whipple, "Computer Pattern Recognition Techniques: Electrocardiographic Diagnosis", Comm. of the ACM. 5, pp. 529-532 (October 1962).

13. Ball, G. H., and D. J. Hall., "Some Fundamental Concepts and Synthesis Procedures for Pattern Recognition Preprocessors", paper presented at the International Conference on Microwaves, Circuit Theory, and Information Theory, September 7-11, 1964, Tokyo, Japan.
14. Nilsson, op. cit., Chapt. 7.
15. Williams, J. D., The Compleat Strategyst, (McGraw-Hill Book Co., New York, N.Y., 1954).
16. Munson, J. H., et al., "A Pattern-Recognition Facility with a Computer-Controlled Learning Machine," paper presented at the 1965 IFIP Congress.
17. Ridgway, W. C., "An Adaptive Logic System with Generalizing Properties," Technical Report 1556-1, Contract AF 33(616)-7776, Stanford Electronics Laboratories, Stanford University, Stanford, California, (April 1962).
18. Novikoff, A. B. J., "On Convergence Proofs for Perceptrons," Technical Report, Contract Nonr 3438(00), Stanford Research Institute, Menlo Park, California, (January 1963).
19. Agmon, S., "The Relaxation Method for Linear Inequalities," Can. J. Math., 6, No. 3, pp. 382-392, (1954).
20. Motzkin, T. S., and I. J. Schoenberg, "The Relaxation Method for Linear Inequalities," Can. J. Math., 6, No. 3, pp. 393-404 (1954).

Technical Documentary Reports
Prepared During the Program

A Committee Solution of the Pattern Recognition Problem

by C. M. Ablow and D. J. Kaylor.

Notes on Classification Capacities

by T. Cover

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Stanford Research Institute Menlo Park, Calif		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP N/A	
3. REPORT TITLE Theoretical and Experimental Investigations in Trainable Pattern-Classifying Systems			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Final Report June 64-June 65			
5. AUTHOR(S) (Last name, first name, initial) Nilsson, Nils J.			
6. REPORT DATE SEPTEMBER 1965		7a. TOTAL NO. OF PAGES 103	7b. NO. OF REFS 20
8a. CONTRACT OR GRANT NO. AF30(602)-3448		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO. 5581			
c. Task #558104		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) RADC-TR-65-257	
d.			
10. AVAILABILITY/LIMITATION NOTICES Available through DDC.			
11. SUPPLEMENTARY NOTES "A Committee Solution of the Pattern Recognition Problem", Ablow & Kaylor. Notes on Classification Capacities. T. Cover.		12. SPONSORING MILITARY ACTIVITY Rome Air Development Center (EMIID) Griffiss AFB, New York 13442	
13. ABSTRACT <p>Motivation is given for the use of various trainable pattern-classifying structures called <u>linear</u> and <u>piecewise linear (PWL)</u> machines. The results of various experiments in training these machines are presented. Two different types of training methods were investigated: mode-seeking and error-correction methods. These methods are illustrated by experiments using two-dimensional patterns so that the results of training can be easily visualized. More thorough experiments with 10-dimensional and 100-dimensional patterns are also described.</p> <p>It is concluded that certain of these training methods can be expected to give good performance in complex pattern classifying tasks involving multi-modal pattern distributions. The report concludes with a list of recommendations for future research, and contains an Appendix presenting a new theorem on training a linear machine.</p>			

DD FORM 1473
1 JAN 64

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT.	ROLE	WT.	ROLE	WT.
Pattern Recognition Linear Separability Non-Linear Separability Piecewise Linear Machines Error Correction Procedures Multi-Modal Pattern Distributions						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U).

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification