1964

A learning matrix, with application to pattern recognition

Thomas Michael Whitney
Iowa State University

A LEARNING MATRIX, WITH APPLICATION TO PATTERN RECOGNITION

by

Thomas Michael Whitney

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

Major Subject: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Head of Major Department

Signature was redacted for privacy.

Dean of Graduate College

Iowa State University
Of Science and Technology
Ames, Iowa

1964

TABLE OF CONTENTS

INTRODUCTION

Simply stated, pattern recognition is the process of partitioning events into a set of mutually exclusive classes. This partitioning may be divided into two separate functions; 1) a "transducer" which senses the patterns to be identified and converts the information acquired into numerical measurements to "represent" the pattern, and 2) a "categorizer" which accepts these signals and by some means interprets them as belonging to a particular class.

Interest is generated in this topic by the wide variety of uses for which a pattern recognition system might be employed. A common example is the need for reading machines to interpret numbers, letters, and symbols for accounting and data processing applications. Banks and post offices could also make valuable use of machines which reliably read cursive handwriting. More complex recognition tasks include the analysis of air or satellite photos for weather forecasts, speech recognition, identification of objects on air photos for military purposes, and interpretation of sampled electrical signals such as electrocardiograms.

The learning matrix is a matrix-like circuit structure which could be used as a categorizer in a pattern recognition system. It is a variable-parameter device in which the intersection points of the matrix rows and columns are formed by adjustable connecting elements called weights. It is also adaptive since the connections which determine its behavior are adjusted during its performance (in real time) to improve or optimize the overall performance of the system. The device is also said to "learn" in the following sense:

1) An experiment, organized into a sequence of identical trials, is performed.

2) Each trial produces some quantifiable output by the device.

3) A relationship "better than" is defined for the outputs.

The learning matrix is said "to learn" since an examination of its outputs over a sequence of trials indicates a trend of improved performance.

A schematic diagram of a learning matrix is shown in Figure 1. For a pattern recognition system the input X is considered to be the output of the transducer portion of the system. The output of the learning network is compared to a desired output Z* and some performance measure determined. On the basis of this measure the adjusting system adapts the learning network.

Not all pattern recognition systems have a learning capability. The significance of learning, or as it is sometimes called, self-organization, has been debated at great length. The question is whether or not an adaptive network possesses any inherent advantage over a fixed-parameter network suitably designed for some recognition task. It is clear that for a specific pattern recognition function the learning ability of any network becomes useless once the network has learned the proper responses. A machine which learns to read the English alphabet and thereafter is required to perform no other task, for example, might well be replaced by a simpler fixed-parameter network. However, if the network is ever required to perform additional tasks, such as reading another language, the learning ability becomes justified. Thus the importance of network learning depends largely upon the requirements and economics of specific practical applications.

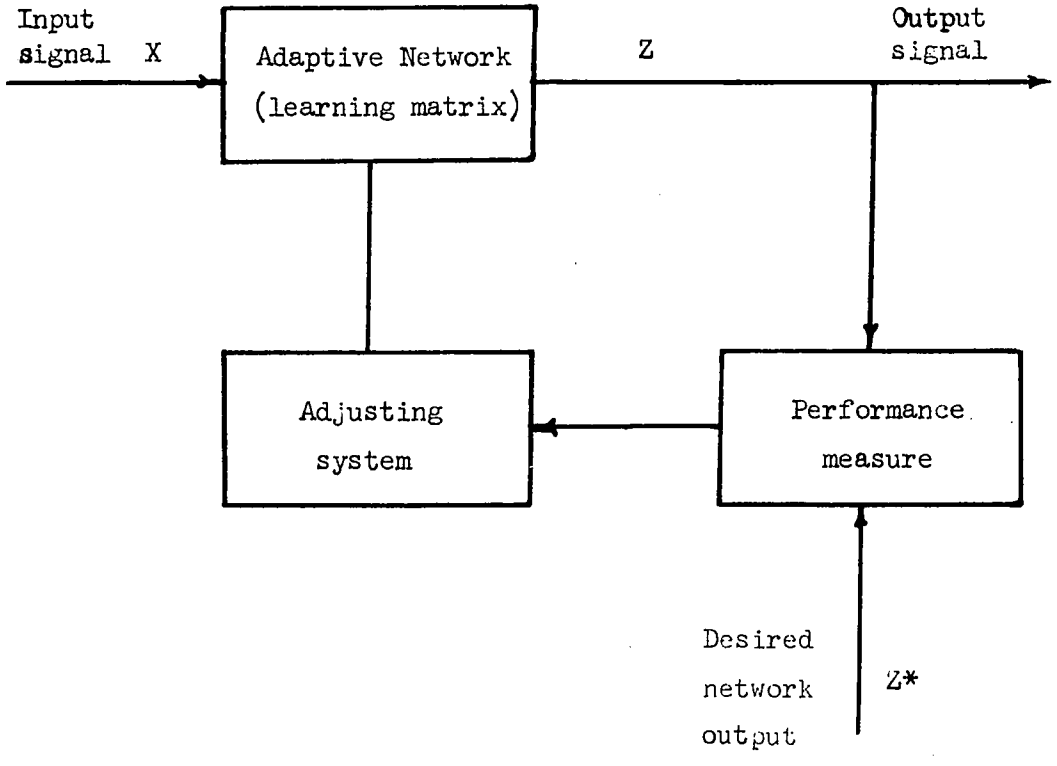


Figure 1. Schematic diagram of an adaptive pattern categorizer

This dissertation provides a primarily theoretical analysis of an adaptive matrix-like circuit structure. This device has direct application to pattern recognition problems. As a further introduction to pattern recognition, a description and classification of various recognition techniques is given in Chapter II. The types of pattern categorizers are divided into three groups depending upon whether the primary principle involved is correlation, linear decision functions, or statistical decision theory. In Chapter III, the details of the learning matrix are described. Two algorithms for training the learning matrix are discussed and theorems related to the convergence of these procedures are proved. A geometrical description of the classification criteria is also provided. Physical elements for possible mechanizations of the learning matrix are discussed in Chapter IV. The most successful of these elements are based on magnetic phenomenon. Chapter V describes the results of a computer simulation study of the learning matrix. Experiments were performed on a set of handwritten and machine printed numbers. Chapter VI includes a summary of the main results of this dissertation and the resulting conclusions which can be drawn. Also mentioned are possible areas of future investigation based on the principle of the learning matrix.

This dissertation considers only the categorization problem of pattern recognition. The other aspect of the problem - that of designing a suitable transducer to extract the essential "features" of a pattern - is in many ways more important to successful pattern classification. For this study, a suitable set of measurements representing a pattern is always assumed to be available. Ideally the features or measurements which describe a pattern should be invariant to position, size, contrast, and

distortion. Any categorizer is never any better than the data with which it is presented from the transducer.

No attention will be given in this study to analogies between the learning matrix and biological neurons or systems. Although several expressions have been borrowed from psychology to describe certain processes, such as learning, forgetting, punishing, and rewarding, attempts at analogy often lead to over-simplification and misunderstanding. Enough factual knowledge of the behavior of nets of natural neurons has not been accumulated to justify a meaningful or useful analogy at this time.

REVIEW OF PATTERN RECOGNITION TECHNIQUES

Introduction

As mentioned in the introduction, the problem of pattern recognition can be divided into two basic functions. First, numerical measurements must be selected which somehow "represent" each pattern, and secondly, a categorizer must be designed which assigns each pattern to the correct class on the basis of these measurements.

The transducer has as its input a physical sample to be recognized and as its output a set of numerical measurements which characterize the input pattern. The output of the transducer constitutes the input to the categorizer which then assigns this input to one of a finite number of categories. The measurements which a transducer makes may be either continuous or discrete. The categorizer applies some type of decision criterion to its input to decide to which category, if any, the input belongs. If the decision is suspected of being questionable or unreliable in some sense, the categorizer may reject the pattern as belonging to none of the categories. The input has then been rejected. If the categorizer attempts a decision and is wrong, it is said an error has been made.

Since mathematical methods cannot deal with electrical signals, physical objects, or optical images directly, a mathematical model of the pattern recognition problem must be constructed. A convenient technique is to let the signals, images, or events to be recognized be represented by points or vectors in an n-dimensional space. Each dimension expresses a property of the event. For instance a pattern could be represented as a vector by superimposing a set of grid squares over the pattern. Each square represents one dimension of the vector. The presence or absence

of a portion of the pattern in each square of the matrix could be represented by a binary one or zero. The entire pattern would be represented by a vector, $X = (x_1, x_2, \ldots, x_n)$, the coordinates of which designate the presence or absence of the pattern in each square. In general the coordinates could have analog values, corresponding to the "amount" of each property present.

The functions of the transducer and the categorizer can be thought of in terms of transformations in R^n, the n-dimensional space. The transducer transforms each pattern into a discrete point in R^n. The categorizer then partitions R^n into disjoint sets of points, A, B, ..., K. Figure 2 shows the relative function of the transducer and categorizer portions of a pattern recognition system.

A pattern recognition system depends upon the successful performance of both the transducer and the categorizer. The transducer must extract a set of features from the pattern which are reliable but not overly redundant. It would be expected that the transducer would tend to cluster vectors from a specific class close together in R^n, in the sense that average distances between vectors from the same class would be small. A simple set of features was described earlier; that of using a binary matrix to represent a pattern. More sophisticated methods used in character recognition (reading of alphanumeric data) use features which consist of shape criteria such as straight lines, slanted lines, cusps opening in various direction, circles, and line intersections. Ideal features would be invariant to all types of "noise" such as size, skew, and deformation differences. One of the major problems of pattern recognition systems is the selection of adequate feature generating transducers.

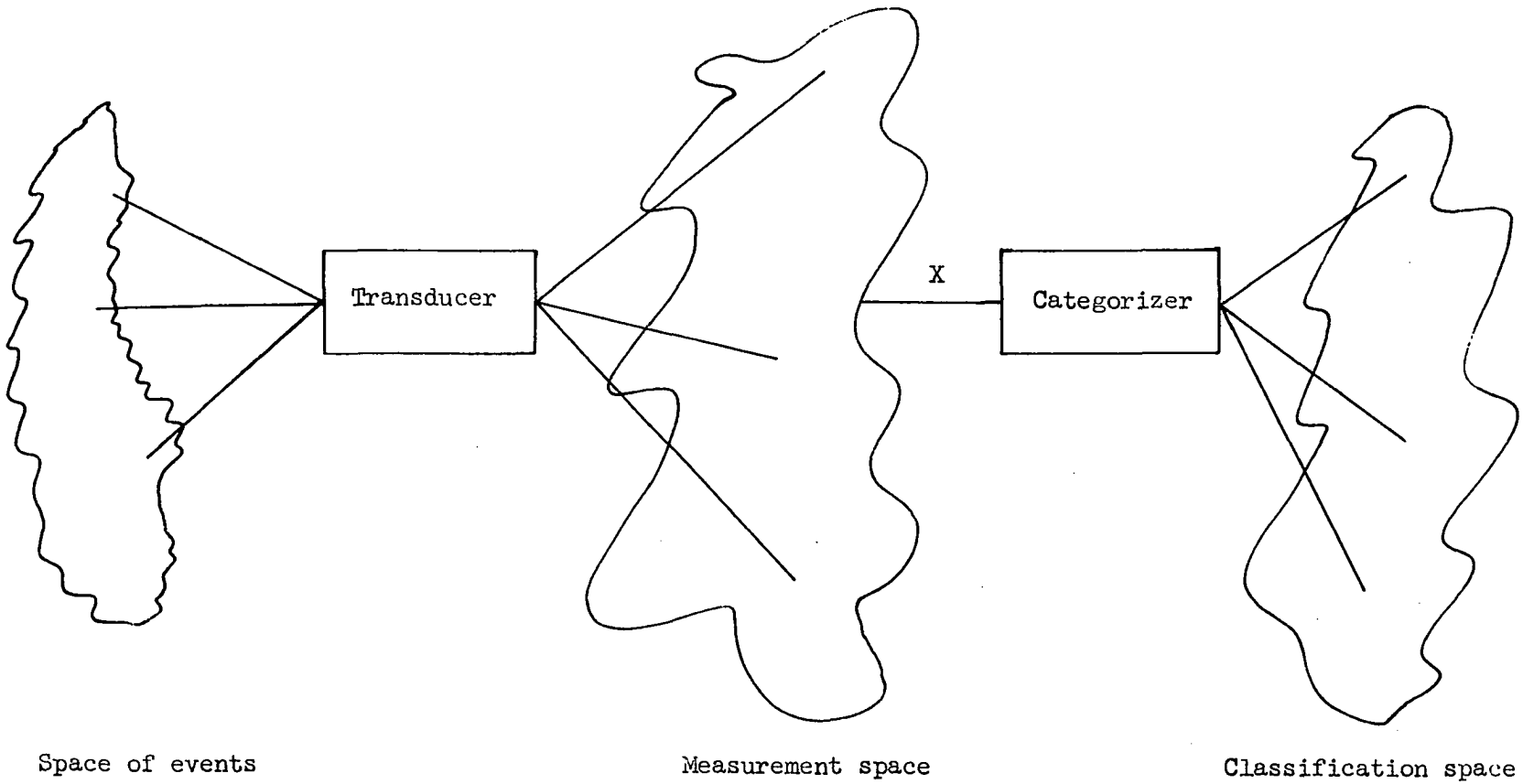


Figure 2. Functions of the transducer and the categorizer

The categorizer is expected to separate the vectors generated by the transducer into disjoint sets. A great many techniques have been devised which perform this function and some of the main types are discussed later in this chapter.

There are two basic types of categorizers. In one type, called a *fixed-parameter system*, the classification technique is determined from a given set of input samples and does not change after the original design. Most of the present systems available for character recognition are of this type. In the second type, called adaptive systems or learning networks, the system parameters are sequentially changed as the repeated occurrence of input and desired output data are presented. Such systems go through a learning cycle, called a training procedure, at the termination of which the device is ready to perform the desired pattern recognition. This differs from the fixed-parameter system in that the adaptive network may be trained to perform a new recognition task by simply undergoing a new training cycle.

For the purpose of the following discussion, the existing literature on pattern classifiers may be divided into three sections: correlation or "mask" techniques, hyperplane or linear decision function techniques, and statistical decision models. Although various other related ideas have been advanced, in most cases existing ideas may be placed in these categories.

The areas also tend to overlap so that occasionally a technique possessing properties of one class can be shown to be equivalent to that of another class. For instance the learning matrix of major concern in this study could be considered either a correlation method or a hyper-

plane method depending upon the training procedure and the point of view.

Correlation Coefficients

One of the simplest decision making methods is to correlate the input vector to be classified with each of several stored references that represent the different classes to which the input may belong. Most existing character readers operate in this way. The stored references or class vectors, W, are often the means of the set of sample vectors of each input class. Decisions are made by comparing the correlation between the input and each of the stored references and by deciding that the input is a member of the class corresponding to the largest correlation coefficient. In terms of the vector space R^n, this amounts to finding the maximum $X \cdot W_i$,[1] where i is taken over all the possible stored references. The shortcomings of this method are illustrated by considering the two classes shown in two dimensional space in Figure 3a. Since correlation computes a dot product, it is a measurement of the cosine of the angle between the reference and the input X. If α is the angle between W_A and X and β the angle between W_B and X, the boundary between the region where X is classed in A and where it is classed in B is a straight line which passes through the origin and causes $|W_A| \cos \alpha = |W_B| \cos \beta$. No matter how W_B is determined from the members of B it is easy to envision situations where perfect classification by this method would be impossible. In more than two dimensions the separating regions would be hyperplanes rather than straight lines. Points that would be misclassified are shown in Figure 3a

---

[1] The dot refers to the dot product between vectors.

by the shaded regions.

If the correlation scheme is designed so that $|W_A| = |W_B|$ by some method of normalization, the decision region boundary becomes the bisector of the angle between W_A and W_B. Then the classification is based only on the angular proximity of the input to the stored weighting vectors. It is a useful method if the angular dispersion between members of each class is small compared to the separation between classes.

Another technique essentially the same as correlation is a decision rule based on the minimum Euclidian distance between the input X and the stored references W_i. An input is classified in A, that is $X \epsilon A$, if $|X - W_A| < |X - W_B|$. This rule partitions the space into two regions by the perpendicular bisector of the line connecting W_A and W_B. If $|W_A| = |W_B|$, minimizing the Euclidian distance is equivalent to maximizing the correlation coefficient.

The correlation coefficient technique and Euclidian distance technique define different decision regions when a threshold or discrimination level is used to provide rejection of the input when the decision is not considered reliable for some reason. With correlation, an input may be rejected if $\max X \cdot W_i < d$ for all i, that is, the input is not similar enough to any of the stored references. The regions of space where inputs are classified in each category are shown in Figure 4a as cone-shaped regions with the stored references bisecting the cones.

A threshold may also be employed with a Euclidian distance decision method by rejection of an input if $|X - W_i| > d$ for all stored references. Acceptable regions are then circles in the two dimensional space as shown in Figure 4b.

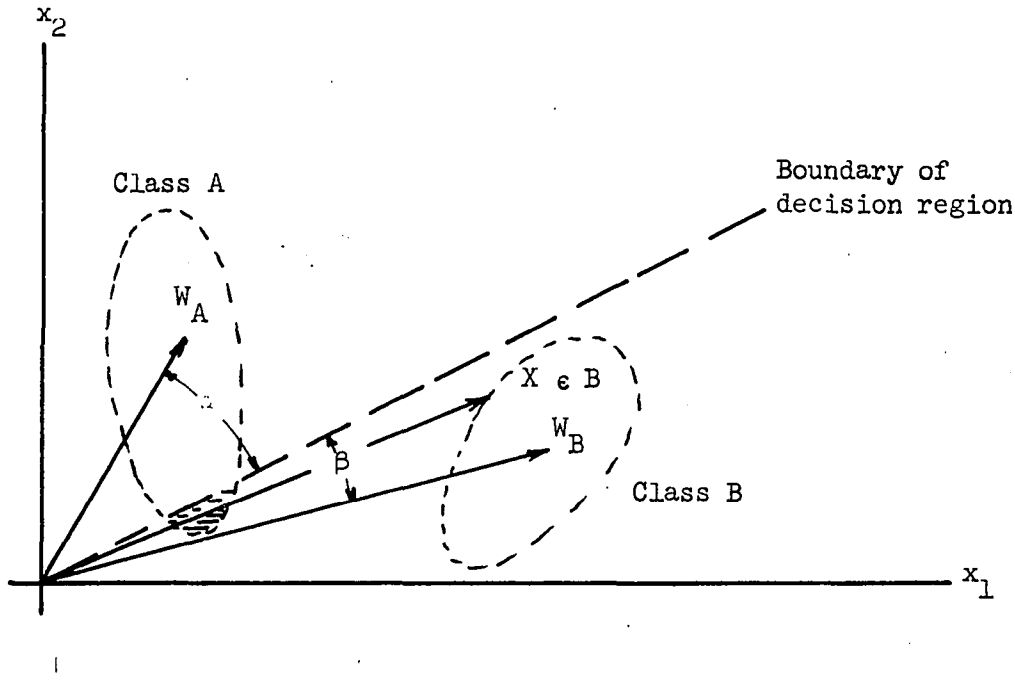


Figure 3a. Classification by maximizing correlation to weight vectors

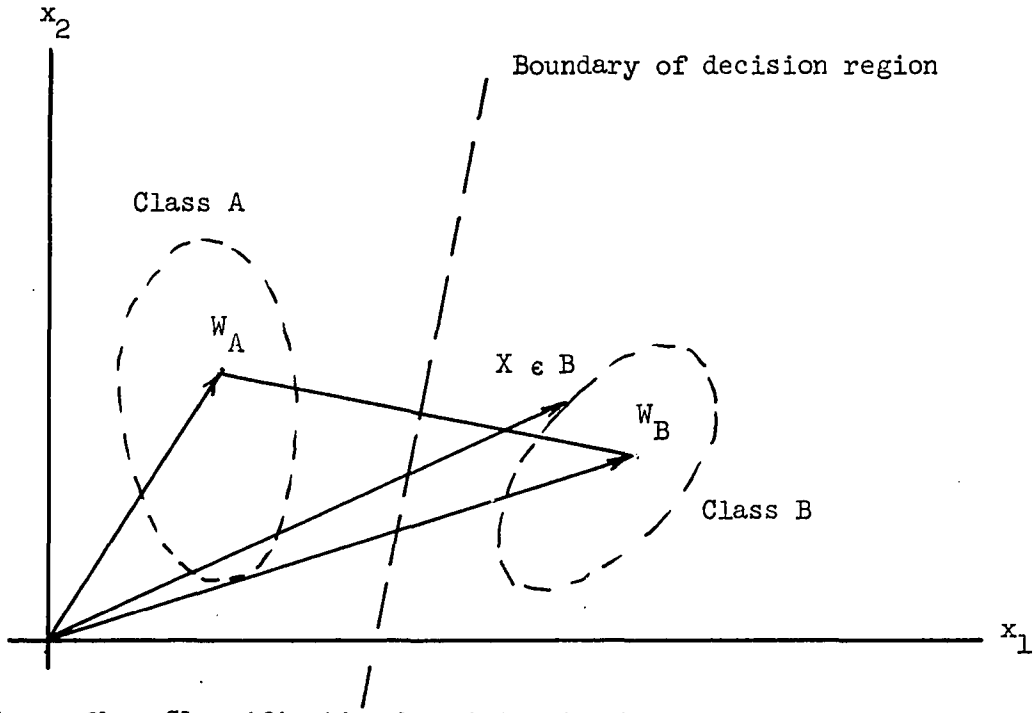


Figure 3b. Classification by minimizing Euclidian distances to weight vectors

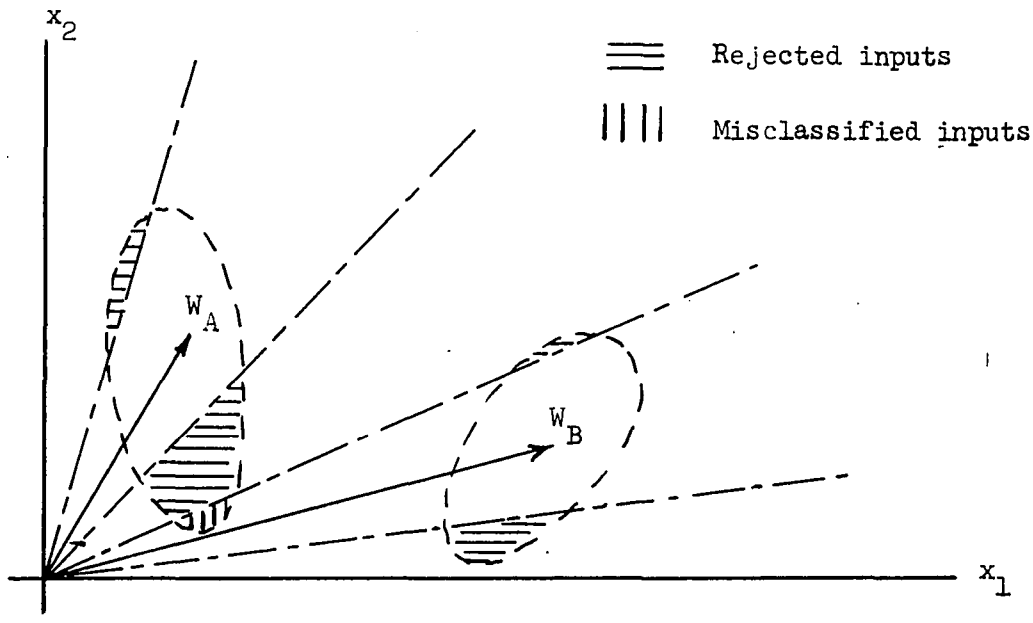


Figure 4a. The use of thresholds with correlation

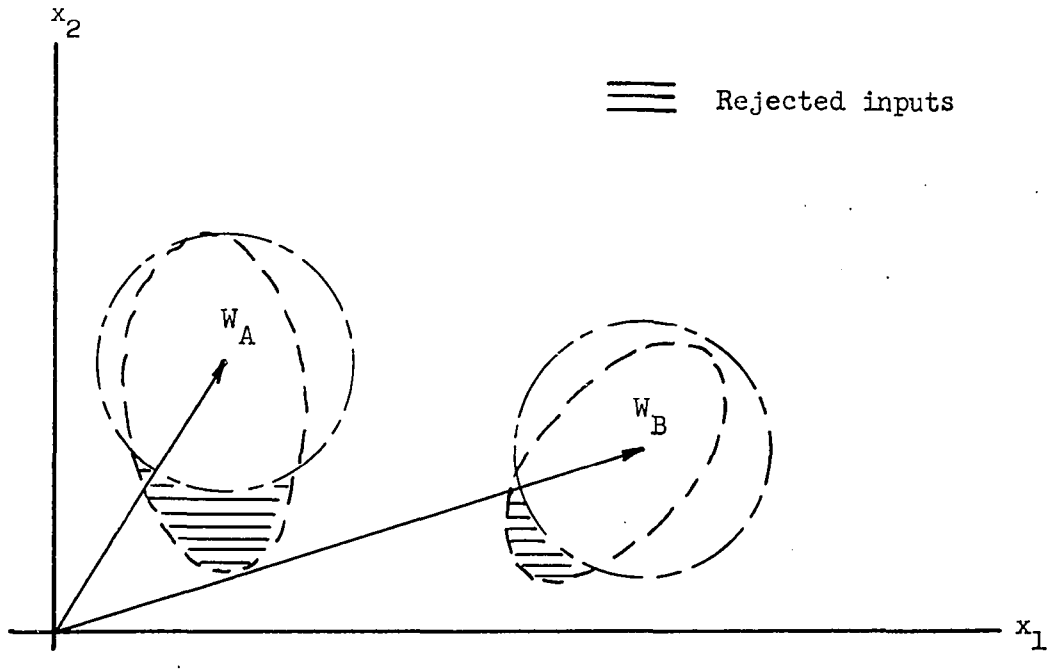


Figure 4b. The use of thresholds with minimum Euclidian distances

A recognition technique based on correlation is described in Highleyman (8).

The stored references of correlation recognition methods may either be precalculated by examining the known members of input sets, or they may be "learned" by a process of sequentially observing an input vector and its known class and adapting the stored reference. A fixed-parameter system would use the first method while a learning network would use the second.

Linear Decision Functions

There is not a great deal of difference between decision processes based on correlation coefficients and the idea of separation of pattern classes by linear decision functions or hyperplanes. As pointed out previously, the boundaries which separate classes using correlation in higher dimensions are hyperplanes which pass through the origin. One hyperplane, defined by a normal vector W and a distance from the origin, d, can be used to separate two classes. All inputs such that $X \cdot W > d$ are placed in one class and if $X \cdot W < d$ the input is placed in the other class. The space is thus separated by this decision process into two half-spaces. The equation $X \cdot W - d = 0$ represents the separating hyperplane. Figure 5 shows a plane in three dimensions which is used to separate the patterns (vectors), (1 1 0), (1 0 1), and (1 1 1) from all other points in space represented by binary inputs. The question of whether or not a hyperplane exists which will perform a specified dicotomy of the binary input space is a major problem of the area of threshold logic and linear separability. A great deal has been published in this

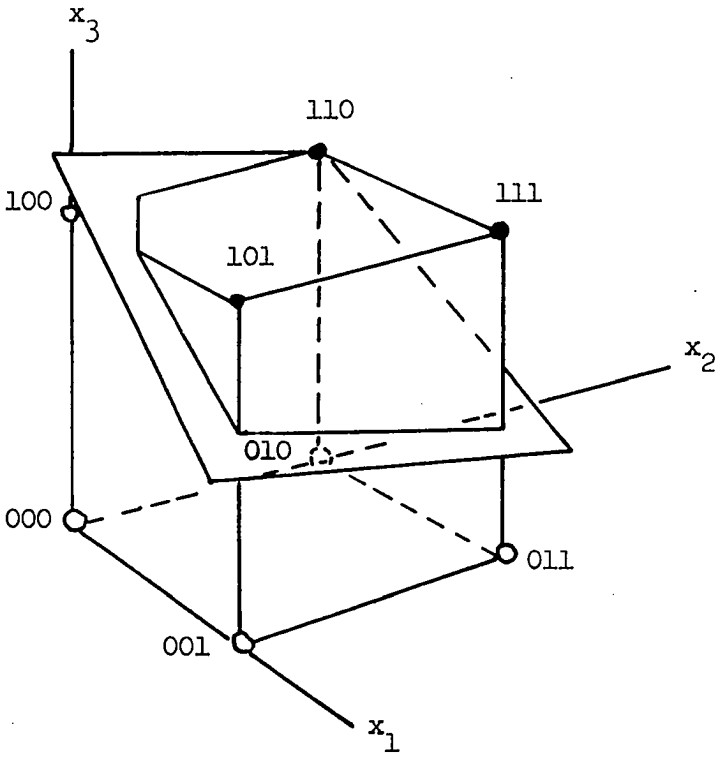


Figure 5. Separation of three-variable space by a hyperplane

area and the interested reader is referred to Mattson (11), Ridgeway (14), and Winder (20), (21) for further information.

One form of pattern recognition using hyperplanes then, is to attempt to find a hyperplane (W_i, d_i) for each pattern class i, such that if $X \epsilon A$, $X \cdot W_A > d_A$, and if $X \not\epsilon A$, $X \cdot W_A < d_A$. That is, to find a hyperplane which separates each class of inputs from all other inputs. If there are k classes, k hyperplanes would be needed. As might be imagined, this imposes some severe restrictions on the location of the sets in the space R^n in order for perfect classification to be possible. In the terminology of convex sets, the requirement for perfect classification can be expressed as follows: The convex hull of each pattern class and the convex hull of all other pattern clases must not intersect. Figure 6a shows the separation of three classes in two dimensional space using one hyperplane per class.

The requirements of the preceeding method can be relaxed somewhat if a hyperplane is used to separate each pair of classes. The decision making process is then reduced to determining on which side of each hyperplane a given input lies. This method is employed by Highleyman(9). With this method the convex hulls of each pair of classes must be non-intersecting for a perfect solution to be possible. This is a much less stringent condition than the one hyperplane per class method. Figure 6b shows the separation of three classes in two dimensions by class pair hyperplanes where the one hyperplane per class technique would not perfectly separate the classes.

The disadvantage of class pair separation is that the required number of hyperplanes increases rapidly. For k classes the number of hyper-

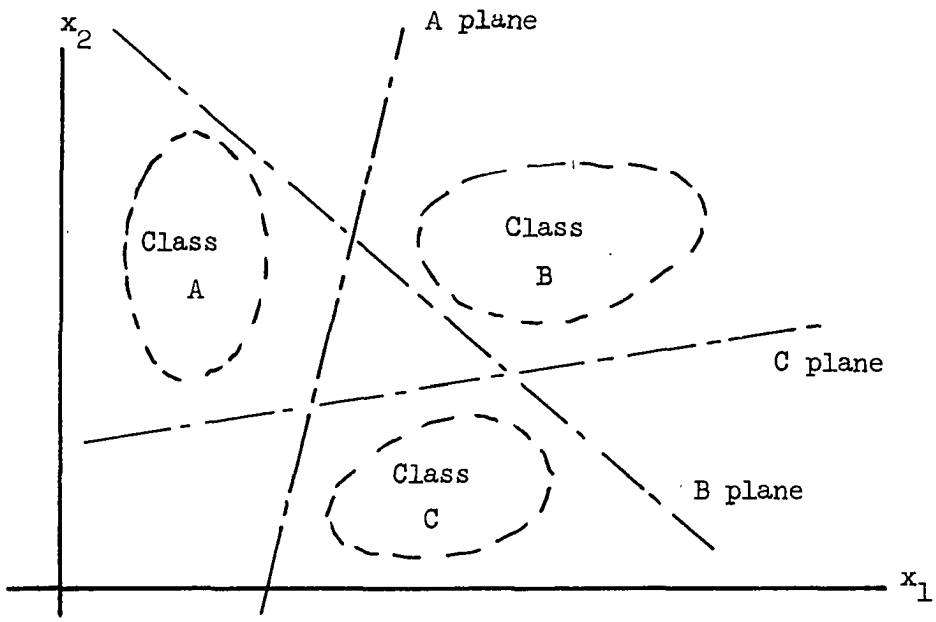


Figure 6a. Class separation by one hyperplane per class

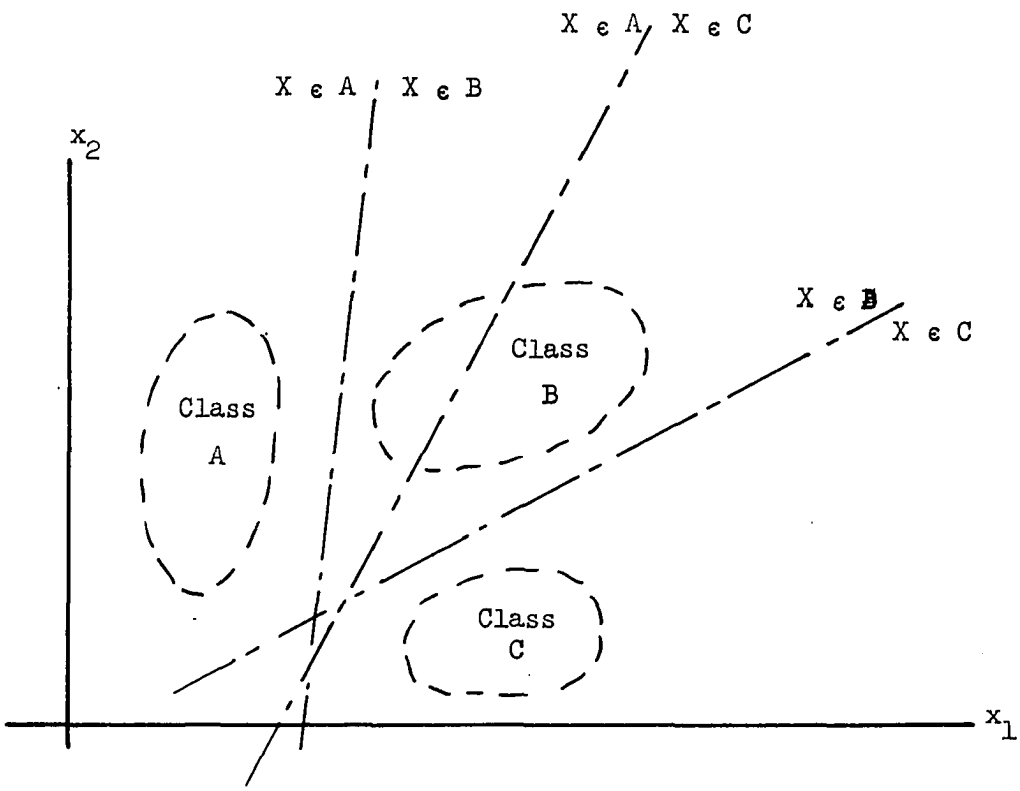


Figure 6b. Class separation by one hyperplane per pair of classes

planes needed is given by[1]

$$\binom{k}{2} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2}.$$

Thus in a character recognition system which recognizes the ten numbers zero through nine, a total of 45 hyperplanes are required for class pair separation whereas only 10 are required for class separation.

It is shown in Highleyman (9) that for any set of data, a pattern recognition method based on linear decision functions (or hyperplanes) is at least as good as any based on minimizing a Euclidian distance or maximizing a cross correlation function.

The idea of a linear weighted sum as a decision process goes back to one of the earliest types of learning networks, the perceptron, proposed by Rosenblatt (16) in 1958. Only later was the geometrical interpretation of hyperplanes associated with the technique. What Rosenblatt succeeded in showing was that a successful set of hyperplanes could always be found by a sequential learning process provided such a set did in fact exist. The proof of the convergence of the learning process was later simplified and amended by others, notably Novikoff (12). The perceptron model is shown in Figure 7. It is assumed that sensory inputs are mapped by means of random connections with fixed weights upon a series of decision units called A-units. No learning occurs in this stage and it may be considered that the output of the A-units represent a transformed input pattern. The transformed inputs are then mapped through variable

---

[1]The exclamation point denotes factorial or, $k! = k(k-1)(k-2)\ldots$

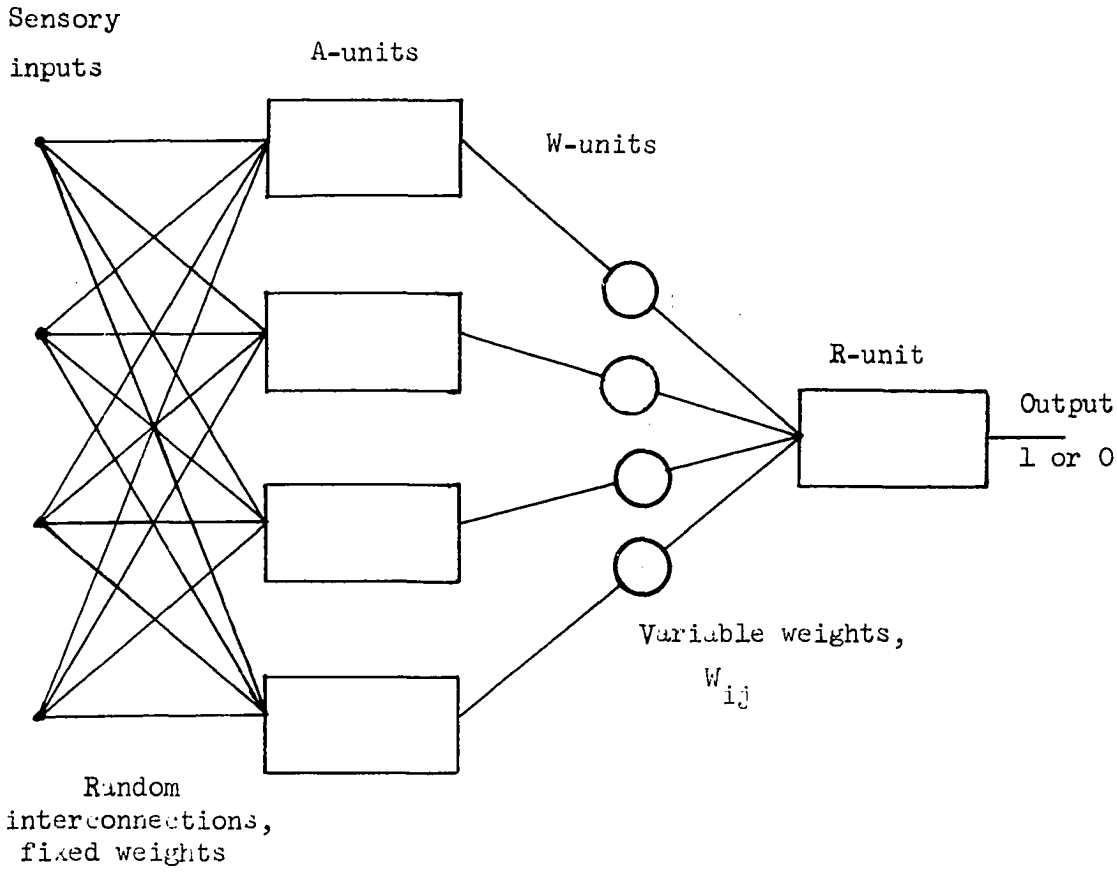


Figure 7. Model of the perceptron

weights to response or R-units (only one shown). The R-unit acts as a threshold element, making a binary decision one if the sum of its inputs is greater than the threshold and zero otherwise. During the learning process the values of the weights w_i are changed whenever the output of the R-unit disagrees with the desired output of the network which must be known during the learning cycle. This process is called reinforced learning.

Thus an advantage of the linear decision function technique is that the designer is assured a learning process will converge to an acceptable set of hyperplanes if such a set exists.

Statistical Decision Theory

The basic function of the categorizers is to make a decision. Attempts to optimize this process have led to pattern recognition techniques based on statistical decision theory. The recognition problem is considered to be a problem of testing multiple hypothesis that a given input is or is not a member of a given class. The necessary information in all statistical decision processes is; (see Harmon (5))

1) The a priori probability of each of the pattern classes or $P(c_i)$,

2) The conditional probability that, given a class c_j, X_i is a member of class c_j, or $P(X_i \mid c_j)$, and

3) The cost associated with making the decision $X_i \in c_j$, or C_{ij}.

Thus C_{ij} is the cost associated with making the decision $X_i \in c_j$ when in fact X was a member of some other class.

This optimum decision process based on the a priori information is called a Bayes' decision rule and is designed so as to make the decision

c_k that minimizes the average cost. In character recognition systems the costs associated with various types of errors are often considered equal. For example there is no greater penalty incurred for calling a five a seven than for calling a five a six; both cases are errors. The Bayes' decision is made by finding through Bayes' Theorem on probability, the conditional probabilities $P(c_j \mid X)$ given by

$$P(c_j \mid X) = \frac{P(X \mid c_j)\, P(c_j)}{\sum_j P(c_j)\, P(X \mid c_j)} . \qquad (1)$$

For the case of two pattern classes, c_1 and c_2, the Bayes' decision rule minimizes the average cost by choosing $X \in c_1$ if

$$\frac{P(X \mid c_1)}{P(X \mid c_2)} > \frac{C_{12} P(c_2)}{C_{21} P(c_1)} = T.$$

Thus the decision rule that minimizes risk compares the ratio of two probability densities with a precalculated threshold value T to decide if X is more likely a member of c_1 or c_2. The ratio of conditional probability densities is called the likelihood ratio and the class of decision rule, Bayes' rules. In the simple case where the costs and a priori probabilities are equal ($T = 1$), a not uncommon situation in pattern recognition, the optimum decision consists of determining the maximum conditional probability $P(c_j \mid X)$ from Equation 1 above. Actually the denominator of the right side of Equation 1 can be removed since it appears for each c_j and does not affect the relative maximum. Figure 8 shows a schematic diagram of an optimum decision network.

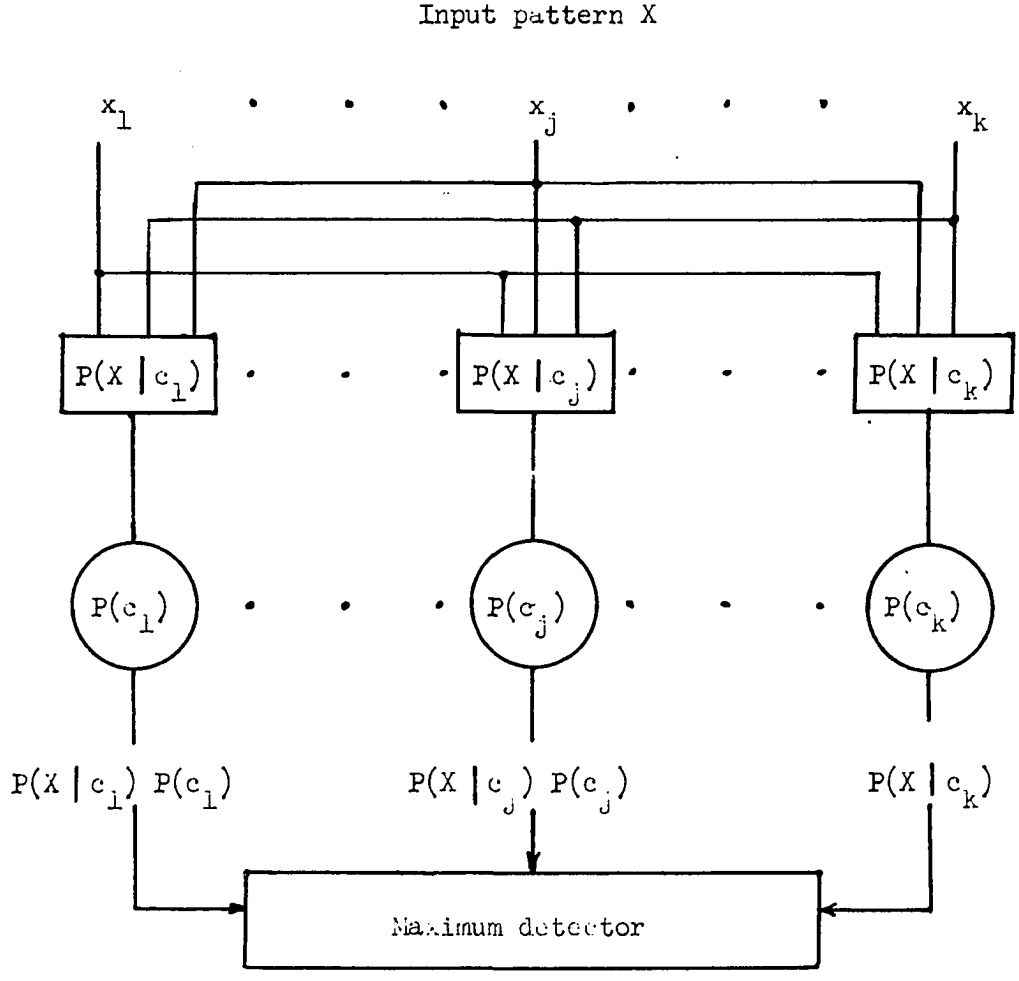


Figure 8. Optimum decision network when costs are equal

A simplification occurs for the statistical decision method if each of the features x_i is assumed to be independent and also binary valued. The conditional probability $P(X \mid c_j)$ is then simply the product of the n individual feature probabilities, or

$$P(X \mid c_j) = P(x_1, x_2, \ldots x_n \mid c_j) = P(x_1 \mid c_j)\, P(x_2 \mid c_j) \ldots P(x_n \mid c_j).$$

Returning to the two class example with equal costs, Bayes' rule becomes, choose $X \in c_1$ if

$$\frac{P(X \mid c_1)\, P(c_1)}{P(X \mid c_2)\, P(c_2)} > 1$$

or

$$\frac{[P(x_1 \mid c_1)\, P(x_2 \mid c_1) \ldots P(x_n \mid c_1)]\, P(c_1)}{[P(x_1 \mid c_2)\, P(x_2 \mid c_2) \ldots P(x_n \mid c_2)]\, P(c_2)} > 1.$$

Taking the logrithm of both sides gives,

$$\log \frac{\left[\prod_{i=1}^{n} P(x_i \mid c_1)\right] P(c_1)}{\left[\prod_{i=1}^{n} P(x_i \mid c_2)\right] P(c_2)} > 0.$$

or

$$\sum_{i=1}^{n} \log \frac{P(x_i \mid c_1)}{P(x_i \mid c_2)} + \log \frac{P(c_1)}{P(c_2)} > 0. \qquad (2)$$

Now since x_i is assumed to be binary, i.e., zero or one, each of the terms, $\log (P(x_i \mid c_1)/P(x_i \mid c_2))$ in Equation 2, can assume only two values. By letting

$$u_i = \log \frac{P(0 \mid c_1)}{P(0 \mid c_2)}, \quad v_i = \log \frac{P(1 \mid c_1)\, P(0 \mid c_2)}{P(0 \mid c_1)\, P(1 \mid c_2)}, \text{ and } K = \log \frac{P(c_1)}{P(c_2)},$$

Equation 2 can be written in the form

$$\sum_{i=1}^{n} v_i x_i + u_i + K > 0. \tag{3}$$

Equation 3 is the condition for choosing an input X to be a member of class c_1. The significance of the equation is that the Bayes' decision rule in these special circumstances is identical to a weighted sum method of pattern recognition as used in a simple threshold element.

An important facet of the Bayes' decision procedure is that the numerical values associated with costs and prior probabilities do not influence the nature of the computations performed on the input. They merely affect the threshold of comparison, T. It might also be noted that besides problems in pattern recognition, many other decision problems are formulated in terms of statistical decision theory. For instance a two class problem might consist of signal present or no signal present and the problem becomes the optimum detection of signals in noise.

There are two factors which severely limit the usefulness of a statistical decision method for pattern recognition. Although such a method is theoretically optimum, these two conditions usually force the employment of non-optimum methods.

The first condition is the assumption that the conditional joint probability densities, $P(X \mid c_j)$, are known. In some cases this may indeed be the case, particularly if the physical process that generates the classes of signals are known. More often than not, however, no knowledge of conditional probabilities is known. In principle, the probability densities can be obtained by observing the relative frequencies with which each of the possible combinations of features of the inputs occurs as the number of samples of the class on which the observations are made approaches infinity. Unfortunately most pattern recognition problems do not operate in this fashion. Most often the designer has to be satisfied with a finite and fairly small number of samples from which to estimate the unknown densities.

A second factor that limits the usefulness of statistical decision theory is the constraints imposed on the realization of the decision rule. If the probability densities are not analytically expressible even if they are known, their values at each point must be stored and tabulated. The resulting storage requirements prohibit the realization of likelihood ratios in all but the simplest cases. One way around this difficulty is to assume a common form for the probability distribution, usually the gaussian distribution. A significant reduction in storage capacity is also obtained by assuming the features or dimensions of each vector are statistically independent. Here only the values $P(x_i \mid c_j)$ have to be tabulated instead of all of the $P(X \mid c_j)$.

Further reviews of pattern recognition methods with extensive bibliographies are given in Hawkins (7), and Sebestyen (17). Not mentioned

here is the separation of pattern classes by higher order surfaces as described in Greenberg and Konheim (4).

THE LEARNING MATRIX

Introduction

The learning matrix is an adaptive network which can serve as a categorizer for a pattern recognition system. The development considered in this study is suggested by a model presented by Pohm, Allen, and Nilsson, (13), called a linear adaptive decision network. A description of an adaptive array called a learning matrix is given in Steinbuch (18).

The matrix accepts as its input a set of analog or binary measurements, $X = (x_1, x_2, \ldots, x_n)$, which represent some type of pattern. The output columns of the matrix designate the pattern class code c_j to which X belongs. If there are k pattern classes the matrix has k output lines; hence the class statements are given in a one out of k code. If the n-dimensional input vectors are to be classified into k classes, the learning matrix is an n x k array.

Figure 9 shows a learning matrix with a three-dimensional input and three output lines. The connections between input and output lines are variable elements called weights and they represent the adaptive feature of the network. One way to describe these connections is to consider that signals are transmitted from input to output by transformer action. Each connection is an independent, variable, coefficient of coupling between transformers. Thus the strength of a given connection may vary continuously between -1 and 1.

To perform a pattern recognition function the operation of the matrix proceeds in two phases, a training phase and a recognition phase. During the training phase the matrix receives an input X together with the pattern class code c_j to which X belongs. The connections to the output

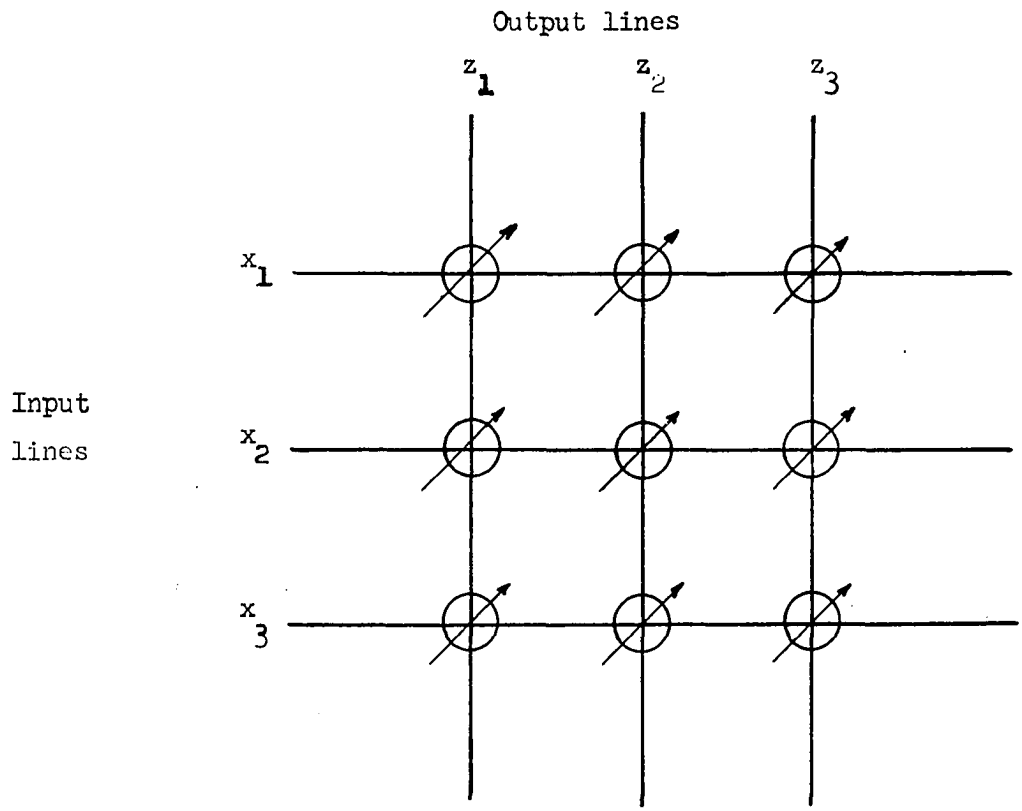


Figure 9. A learning matrix to categorize three dimensional input vectors into three output classes

column for this class are then strengthened or "reinforced" and all other connections to output columns weakened or "punished". After several presentations or iterations of a set of input vectors called an organizing set or training set, the matrix is ready for operation in the recognition mode. New patterns are presented and classified by the matrix on the basis of what it has "learned", i.e., the information it has stored in the variable connections during the training cycle. If these new patterns are not classified correctly, an improvement may be sought by repeating the training phase.

Of the pattern recognition systems described in Chapter II, the learning matrix resembles most clearly the correlation coefficient type. This is because each pattern class is associated with one column of the matrix or one "weighting vector". It can be shown however, that with a particular training routine and method of detecting the class code c_j, the learning matrix becomes analogous to separation of classes in the n-dimensional space R^n by hyperplanes or linear decision functions. These similarities are discussed in detail later in this chapter.

The Continuous Correction Algorithm

The first training procedure or algorithm for adapting the learning matrix classifies patterns by attempting to produce an output of one on the output line associated with the class of each input pattern and a zero on all other output lines of the matrix. An example problem for the learning matrix of Figure 9 might be to classify the three patterns $X_1 = (1\ 0\ 0)$, $X_2 = (1\ 1\ 0)$, $X_3 = (0\ 1\ 1)$, by producing outputs of $Z_1 = (1\ 0\ 0)$, $Z_2 = (0\ 1\ 0)$, $Z_3 = (0\ 0\ 1)$. The first training algorithm is called the

"continuous correction" algorithm since an adaption is made to the learning matrix after every input. It follows the method proposed by Pohm, Allen, and Nilsson (13), and operates as follows:

1) The initial values of all coefficients of coupling are arbitrary but are assumed to be zero.

2) Each pattern (vector) is applied sequentially to the input of the matrix. The connections to the output column corresponding to the class of the input are adjusted until the column output is unity. The connections to all other columns are adjusted so the column outputs are zero.

Thus a column vector W_1 is sought which is orthogonal to both X_2 and X_3 but for which $X_1 \cdot W_1 = 1$. Note that only the connections on input lines with a binary one are adjusted since input lines with a zero input contribute nothing to the column output.

Note also that for this example and all others in which m patterns are to be divided into m classes, Z*, the desired output matrix, equals I, the identity matrix. In these cases, the ideal weighting matrix can be found directly by matrix manipulations. Assuming the input vectors are linearly independent so that X is nonsingular it is easily seen that[1]

$$X W = Z^* = I$$

$$W = X^{-1}.$$

Thus a weighting matrix to perfectly classify the three inputs exists and is equal to the inverse of the input matrix.

---

[1] X^{-1} denotes the inverse of the matrix X.

It may seen strange to bother with a training algorithm if the desired weighting matrix is already known. However, X^{-1} exists only if the training set is composed of n, linearly-independent, n-dimensional vectors. In many cases there may be more or less than n inputs. Also for a large number of dimensions it is difficult to determine X^{-1}. A computer could do so, but the ultimate utility of the learning matrix is an actual physical network for which the mechanization of finding X^{-1} would be difficult.

The first step in analyzing properties of the training algorithm is to formulate the procedure in an exact mathematical language. This can be done efficiently through the use of matrix terminology.

The input patterns are considered to be n-dimensional vectors or row matrices. In general throughout this dissertation capital letters designate matrices, sets, or vectors and small letters designate scalar quantities. An iteration is defined to be one presentation of an organizing set. The following nomenclature is useful in the mathematical manipulations which follow.

i; the row index. X_i is the i-th member of the organizing set, or the i-th row of the input matrix X.

j; the column index. x_{ij} is the j-th dimension of the i-th input vector.

t; an index representing the number of the iteration.

n; the number of dimensions of each input vector.

m; the number of vectors in the organizing set.

k; the number of output classes.

c_j; the j-th output class.

X; the input matrix. It has m rows and n columns.

X_i; the i-th input vector or the i-th row of X.

W; the weighting or learning matrix. It has n rows and k columns.

W_j; the j-th column of the weighting matrix; or, the weighting vector corresponding to output class c_j.

$W(i,t)$; the weighting matrix after adaption to the i-th input during the t-th iteration.

Z_i; the output row matrix given by $Z_i = X_iW$.

Z_i^*; the desired output row matrix. Z_i^* has a 1 in one position, the position of the class corresponding to the i-th input, and a 0 in all other positions.

E_i; the error row matrix after presentation of the i-th input, given by $E_i = Z_i^* - Z_i$.

I; the identity matrix.

The pattern recognition problem is essentially equivalent to carrying out the matrix multiplication $XW = Z$, where X, W, and Z are as described above. The successful separation of patterns is accomplished if $XW = Z^*$.

The description of the continuous correction algorithm can now be expressed in a mathematical form as follows:

1) $W(0,0) = 0$, the zero matrix.

2) Each input X_i is postmultiplied by the weighting matrix W to give a row output Z_i.

3) An error matrix is generated by subtracting the actual output Z_i from the desired output Z_i^*. That is, $E_i = Z_i^* - Z_i = Z_i^* - X_iW$.

4) The weighting matrix is adapted so that if $X_i \in c_p$, $X_iW_p = 1$ and $X_iW_j = 0$, for all $j \neq p$.

5) The next input, X_{i+1}, is then considered.

It was discovered that 4) could be accomplished if W is changed by adding ΔW to W, where[1]

$$\Delta W = \frac{X_i' E_i}{\left| X_i \right|^2} = \frac{X_i' (Z_i^* - X_i W)}{\left| X_i \right|^2} .$$

Three training iterations are carried out in Table 1 for the example problem mentioned previously where

$$X = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix} \quad \text{and} \quad Z^* = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} .$$

Table 1. Three training iterations of the learning matrix using the continuous correction algorithm. $W(0,0) = 0$.

Iteration t	Matrix	Input X_1	Input X_2	Input X_3
	Z	(0 0 0)	(1 0 0)	(-1/2 1/2 0)
	E	(1 0 0)	(-1 1 0)	(1/2 -1/2 1)
1	ΔW	$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} -1/2 & 1/2 & 0 \\ -1/2 & 1/2 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 \\ 1/4 & -1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$

[1] The prime (') is used to denote the transpose of a matrix.

Table 1. (Continued)

Iteration t	Matrix	Input X_1	Input X_2	Input X_3
	W	$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/2 & 1/2 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$
	Z	$(1/2 \quad 1/2 \quad 0)$	$(3/4 \quad 1/4 \quad 1/2)$	$(-3/8 \quad 3/8 \quad 3/4$
	E	$(1/2 \quad -1/2 \quad 0)$	$(-3/4 \quad 3/4 \quad -1/2)$	$(3/8 \quad -3/8 \quad 1/4$
2	ΔW	$\begin{vmatrix} 1/2 & -1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} -3/8 & 3/8 & -1/4 \\ -3/8 & 3/8 & -1/4 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 \\ 3/16 & -3/16 & 1/8 \\ 3/16 & -3/16 & 1/8 \end{vmatrix}$
	W	$\begin{vmatrix} 1 & 0 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$	$\begin{vmatrix} 5/8 & 3/8 & -1/4 \\ -5/8 & 5/8 & 1/4 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$	$\begin{vmatrix} 5/8 & 3/8 & -1/4 \\ -7/16 & 7/16 & 3/8 \\ 7/16 & -7/16 & 5/8 \end{vmatrix}$
	Z	$(5/8 \quad 3/8 \quad -1/4)$	$(9/16 \quad 7/16 \quad 3/8)$	$(-9/32 \quad 9/32 \quad 13/16)$
	E	$(3/8 \quad -3/8 \quad 1/4)$	$(-9/16 \quad 9/16 \quad -3/8)$	$(9/32 \quad -9/32 \quad 3/16)$
3	ΔW	$\begin{vmatrix} 3/8 & -3/8 & 1/4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} -9/32 & 9/32 & -3/16 \\ -9/32 & 9/32 & -3/16 \\ 0 & 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 & 0 \\ 9/64 & -9/64 & 3/32 \\ 9/64 & -9/64 & 3/16 \end{vmatrix}$
	W	$\begin{vmatrix} 1 & 0 & 0 \\ -7/16 & 7/16 & 3/8 \\ 7/16 & -7/16 & 5/8 \end{vmatrix}$	$\begin{vmatrix} 23/32 & 9/32 & -3/16 \\ -23/32 & 23/32 & 3/16 \\ 7/16 & -7/16 & 5/8 \end{vmatrix}$	$\begin{vmatrix} 23/32 & 9/32 & -3/16 \\ -37/64 & 37/64 & 9/32 \\ 37/64 & -37/64 & 23/32 \end{vmatrix}$

The question now arises, "Does the training algorithm described, in fact cause the learning matrix to converge to X^{-1}?" The value of the

learning matrix after each of three iterations is shown in Table 1. $W(3,3)$ and X^{-1} are also shown below.

$$W(3,3) = \begin{vmatrix} 23/32 & 9/32 & -3/16 \\ -37/64 & 37/64 & 9/32 \\ 37/64 & -37/64 & 23/32 \end{vmatrix} \qquad X^{-1} = \begin{vmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & 1 \end{vmatrix}$$

It appears that the learning matrix is converging to the desired matrix X^{-1}. But the appearance of convergence for <u>any</u> number of examples (and many have been worked out) is not of particular value. What is needed for this algorithm is a <u>proof</u> of convergence under general conditions for input vectors and output classes. This problem is solved in the following section.

Theorems and Proofs

A mathematical description of the continuous correction algorithm has been discussed previously. A general expression for the adaption process is given by

$$W(i,t) = W(i - 1, t) + \frac{X_i' E_i}{\left|X_i\right|^2}$$

$$W(i,t) = W(i - 1, t) + \frac{X_i'(Z_i^* - X_i W)}{\left|X_i\right|^2} . \qquad (4)$$

With these preliminaries, an equation for the weighting matrix after t iterations can be derived as a function of the input vectors and the

desired outputs. This is stated formally in Theorem 1.

Theorem 1. An equation for the weighting matrix after t iterations, W(t), can be expressed as a function of the organizing set, X, and the desired output matrix, Z*.

Proof. From Equation 4, W(i,t) is expressed in terms of W(i - 1, t). Thus the weighting matrix at any stage of learning is always a function of the present input vector and desired output, and the preceeding weighting matrix. The following development is for an organizing set containing three vectors, although an identical procedure would be followed in a proof for any number of inputs.

The weighting matrix after adaption to the third input during the t-th iteration is

$$W(3,t) = W(2,t) + \frac{X_3' \left[Z_3^* - X_3 W(2,t)\right]}{\left|X_3\right|^2}$$

$$= \left[I - \frac{X_3'X_3}{\left|X_3\right|^2}\right] W(2,t) + \frac{X_3'Z_3^*}{\left|X_3\right|^2} . \qquad (5)$$

Similarly, W(2,t) and W(1,t) can be written as

$$W(2,t) = \left[I - \frac{X_2'X_2}{\left|X_2\right|^2}\right] W(1,t) + \frac{X_2'Z_2^*}{\left|X_2\right|^2} \qquad (6)$$

and

$$W(1,t) = \left[I - \frac{X_1'X_1}{|X_1|^2} \right] W(3,t-1) + \frac{X_1'Z_1^*}{|X_1|^2} . \tag{7}$$

Combining Equation 5, 6, and 7 to eliminate W(1,t) and W(2,t) gives

$$\begin{aligned} W(3,t) = \Bigg[I &- \frac{X_1'X_1}{|X_1|^2} - \frac{X_2'X_2}{|X_2|^2} - \frac{X_3'X_3}{|X_3|^2} + \frac{X_2'X_2X_1'X_1}{|X_1|^2|X_2|^2} \\ &+ \frac{X_3'X_3X_1'X_1}{|X_3|^2|X_1|^2} + \frac{X_3'X_3X_2'X_2}{|X_3|^2|X_2|^2} - \frac{X_3'X_3X_2'X_2X_1'X_1}{|X_3|^2|X_2|^2|X_1|^2} \Bigg] W(3,t-1) \\ &+ \frac{X_1'Z_1^*}{|X_1|^2} + \frac{X_2'Z_2^*}{|X_2|^2} + \frac{X_3'Z_3^*}{|X_3|^2} - \frac{X_2'X_2X_1'Z_1^*}{|X_2|^2|X_1|^2} \\ &- \frac{X_3'X_3X_1'Z_1^*}{|X_3|^2|X_1|^2} - \frac{X_3'X_3X_2'Z_2^*}{|X_3|^2|X_2|^2} + \frac{X_3'X_3X_2'X_2X_1'Z_1^*}{|X_3|^2|X_2|^2|X_1|^2} . \end{aligned} \tag{8}$$

From Equation 8, W(3,t) can be expressed compactly as

$$W(3,t) = A\, W(3,t-1) + B, \tag{9}$$

where A is the entire matrix coefficient of W(3,t - 1) in Equation 8. It is the sum of eight 3 x 3 matrices and for the general case of m, n-dimen-

sional vectors, is a square n by n matrix. The matrix B in Equation 9 is the sum of the remaining matrix terms in Equation 8. In general it is an n by k matrix where k is the number of output classes. The significance of Equation 9 is that both A and B depend only upon the input matrix X and the desired output Z* and therefore could be computed immediately in any problem.

In general, for an organizing set of m input vectors, A is given by

$$A = \left[I - \frac{X_m' X_m}{|X_m|^2} \right] \left[I - \frac{X_{m-1}' X_{m-1}}{|X_{m-1}|^2} \right] \cdots \left[I - \frac{X_1' X_1}{|X_1|^2} \right]$$

$$= \prod_{i=1}^{m} \left[I - \frac{X_i' X_i}{|X_i|^2} \right]. \qquad (10)$$

Note A is the sum of 2^m matrices and hence becomes very complex for large m.

Now using the recursive relationship given by Equation 9, an equation for the learning matrix after any number of iterations can be found as stated in Theorem 1. In all cases which follow, the weighting or learning matrix will be expressed in terms of its value after adaption with the last input in the organizing set, in this case, X_3. Thus W(3,t) will be written simply as W(t). From Equation 9, with W(0) = 0,[1]

---

[1] $A^k = \underbrace{A \cdot A \cdot \cdots A}_{k \text{ times}}$, and $A^0 = I$

$$W(1) = A\,W(0) + B = B$$

$$W(2) = A\,W(1) + B = (A + I)\,B$$

$$W(3) = A\,W(2) + B = (A^2 + A + I)\,B.$$

In general,

$$W(t) = (A^{t-1} + A^{t-2} + \ldots + A + I)\,B = \left[\sum_{k=0}^{t-1} A^k\right] B. \qquad (11)$$

Equation 11 is thus a proof of Theorem 1.

To check the validity of this relationship the example problem worked out in Table 1 is repeated using Equation 11. To simplify writing the equation for A let

$$Q_i = \frac{X_i' X_i}{\left|X_i\right|^2}.$$

The expression for A with three inputs then becomes

$$A = I - Q_1 - Q_2 - Q_3 + Q_2Q_1 + Q_3Q_1 + Q_3Q_2 - Q_3Q_2Q_1.$$

$$= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} - \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 0 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \end{vmatrix}$$

$$+ \begin{vmatrix} 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 1/4 & 1/4 & 0 \\ 1/4 & 1/4 & 0 \end{vmatrix} - \begin{vmatrix} 0 & 0 & 0 \\ 1/4 & 0 & 0 \\ 1/4 & 0 & 0 \end{vmatrix}$$

$$A = \begin{vmatrix} 0 & -1/2 & 0 \\ 0 & 1/4 & -1/2 \\ 0 & -1/4 & 1/2 \end{vmatrix} .$$

To simplify writing B let

$$P_i = \frac{X_i' Z_i^*}{\left| X_i \right|^2} .$$

Then the expression for B with three output classes and three inputs is

$$B = P_1 + P_2 + P_3 - Q_2P_1 - Q_3P_1 - Q_3P_2 + Q_3Q_2P_1.$$

$$= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 1/2 \\ 0 & 0 & 1/2 \end{vmatrix} - \begin{vmatrix} 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

$$- \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} - \begin{vmatrix} 0 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 1/4 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 1/4 & 0 & 0 \\ 1/4 & 0 & 0 \end{vmatrix} = \begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix} .$$

Now using A and B, W(t) can be computed.

$$W(1) = B$$

The computation of W(3,1) in Table 1 checks with B as calculated above.

$$W(2) = (A + I)\, B$$

$$= \begin{vmatrix} 1 & -1/2 & 0 \\ 0 & 5/4 & -1/2 \\ 0 & -1/4 & 3/2 \end{vmatrix} \begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$$

$$= \begin{vmatrix} 5/8 & 3/8 & -1/4 \\ -7/16 & 7/16 & 3/8 \\ 7/16 & -7/16 & 5/8 \end{vmatrix}$$

This result checks W(3,2) as computed in Table 1.

There is as yet no proof that the matrix series given by Equation 11 converges as t approaches infinity. The problem is to find

$$W_{final} = \lim_{t \to \infty} W(t) = \left[\sum_{t=0}^{\infty} A^t \right] B. \qquad (12)$$

To prove that Equation 12 above converges, it is necessary to prove that the infinite matrix series, $\sum_{t=0}^{\infty} A^t$ converges. A theorem from Varga (19) provides necessary and sufficient conditions that a matrix power series converge.

Theorem 2. (Varga) If A is a complex n by n matrix, then A is convergent if and only if $\rho(A) < 1$, where $\rho(A)$ is the maximum eigenvalue of A.

Proof. Only the general idea of the proof is given here. It consists of reducing A to its Jordan normal form by finding a matrix S such that $SAS^{-1} = J$ where J is the Jordan normal form of A. Then J has eigenvalues λ_i on the diagonal and ones or zeros on the superdiagonal. Taking J^t

raises λ^t so that for the matrix series to converge the maximum λ must be less than one. A rigorour proof of this is given in Varga (19).

Thus in order for Equation 12 to converge it must be shown that the maximum eigenvalue of A is less than one in absolute value. The proof of this is one of the more important aspects of this thesis. The following lemma is useful in proving this result.

Lemma 1. For any n-dimensional vectors Y and X,

$$\left| Y(I - X'X) \right| \leq \left| Y \right| .$$

Proof. Let $e_1, e_2, \ldots, X$ be an orthonormal basis for the n-dimensional space. Then any vector Y can be expressed as

$$Y = c_1 e_1 + c_2 e_2 + \ldots + c_n X.$$

and

$$\left| Y \right| = [c_1^2 + c_2^2 + \ldots + c_n^2]^{1/2}.$$

Hence,

$$\left| Y(I - X'X) \right| = (c_1 e_1 + \ldots + c_n X)(I - X'X)$$

$$= [c_1^2 + \ldots + c_{n-1}^2]^{1/2}.$$

Thus,

$$\left| Y(I - X'X) \right| \leq \left[c_1^2 + \ldots + c_n^2 \right]^{1/2} = \left| Y \right|$$

and the lemma is proved. The equality holds only if $c_n = 0$, i.e. Y is perpendicular to X.

Theorem 3. Consider an organizing set composed of n, linearly independent, normalized,[1] n-dimensional vectors. Then all eigenvalues of the matrix A are less than 1 in absolute value, where $A=(I-X_n'X_n)(I-X_{n-1}'X_{n-1}) \ldots (I-X_1'X_1)$.

Proof. Let $e_1, e_2, \ldots, X_n$ be an orthonormal basis for the n-dimensional space. Let Y be any non-zero n-dimensional vector. Then, applying Lemma 1, $\left| Y(I - X_n'X_n) \right| \leq \left| Y \right|$. Now choose a new basis for the space, one vector of which is X_{n-1}. The vector $Y(I - X_n'X_n)$ can be expressed in terms of this new basis. Again applying Lemma 1,

$$\left| Y(I - X_n'X_n)(I - X_{n-1}'X_{n-1}) \right| \leq \left| Y(I - X_n'X_n) \right| \leq \left| Y \right|$$

Repeated application of Lemma 1 then gives,

$$\left| Y(I - X_n'X_n) \ldots (I - X_1'X_1) \right| = \left| YA \right| \leq \left| Y \right| .$$

In particular, if Y is any eigenvector of A,

$$\left| YA \right| = \left| \lambda Y \right| = \left| \lambda \right| \left| Y \right| \leq \left| Y \right| .$$

---

[1] No loss of generality is involved in assuming the input vectors to be normalized since Equation 10 shows the vectors to be normalized anyway by the iterative procedure. The only reason for assuming them normalized here is to reduce the complexity of the mathematical expressions.

Hence,

$$|\lambda| \leq 1.$$

To prove Theorem 3, it must be shown that the equality $\lambda = 1$ cannot hold. For the equality to hold,

$$|YA| = |Y(I - X_n'X_n) \ldots (I - X_1'X_1)| = |\lambda Y| .$$

The transformation A may be thought of as a series of transformations $(I - X_i'X_i)$. It is shown in Lemma 1 that the effect of each of these transformations on any vector is to either reduce the magnitude of the vector or to leave the magnitude unchanged. For the equality to hold for the entire transformation A, it must hold for <u>each step</u> of the transformation, for if the magnitude of the original vector is reduced it can never be increased. Thus for the equality to hold for any non-zero vector, Y, the following equations must hold.

$$|Y(I - X_n'X_n)| = |Y|$$

$$|Y(I - X_n'X_n)(I - X_{n-1}'X_{n-1})| = |Y(I - X_n'X_n)|$$

$$\vdots$$

$$|Y(I - X_n'X_n) \ldots (I - X_1'X_1)| = |Y(I - X_n'X_n) \ldots (I - X_2'X_2)|$$

The first of these equations implies that Y is perpendicular to X_n so that $Y(I - X_n'X_n) = Y$. Using this result, the second equation becomes,

$$\left|Y(I - X_{n-1}'X_{n-1})\right| = \left|Y\right|$$

which implies that Y is also perpendicular to X_{n-1}. Proceeding in this fashion for each of the equations shows finally,

$$\left|Y(I - X_1'X_1)\right| = \left|Y\right|$$

so that Y is also perpendicular to X_1. Thus Y is perpendicular to X_1, X_2, . . ., X_n, a contradiction since an n-dimensional vector Y cannot be simultaneously orthogonal to n independent vectors. Thus the assumption that the equality $\left|YA\right| = \left|Y\right|$ holds was incorrect and therefore $\left|\lambda\right| < 1$, completing the proof.

It has thus been shown that $\sum_{t=0}^{\infty} A^t$ converges. A necessary condition for this convergence is that $\lim_{t\to\infty} A^t = 0$, the null matrix. Using this result it can easily be shown that the initial value of the weighting matrix does not affect the overall convergence of the iterative process. Since

$$W_f = \lim_{t\to\infty} A^t W(0) + \left(\sum_{k=0}^{t} A^k \right) B ,$$

the term of W_f containing W(0) approaches zero as t approaches infinity.

Now that it has been established in Theorem 3 that $\sum_{t=0}^{\infty} A^t$ converges, the next question is, "What does it converge to?" This is stated

formally as Theorem 4.

Theorem 4. The infinite matrix series $\sum_{t=0}^{\infty} A^t$ converges to $(I - A)^{-1}$.

Proof. The fact that the series does converge is established in Theorems 2 and 3. Therefore let

$$S = \sum_{t=0}^{\infty} A^t.$$

Then

$$S - AS = (I - A - A^2 - \ldots) - (A - A^2 - A^3 - \ldots)$$

$$= I$$

$$S(I - A) = I$$

$$S = (I - A)^{-1},$$

thus proving the theorem.

A necessary condition for convergence of $\sum_{t=0}^{\infty} A^t$ can also be established.

Theorem 5. The infinite matrix power series

$$\sum_{k=0}^{\infty} A^k = I + A + A^2 + A^3 + \ldots$$

will converge only if $(I - A)^{-1}$ exists.

Proof. Assume the series converges to some matrix, S, but that $(I - A)^{-1}$ does not exist. From Theorem 4,

$$S\,(I - A) = I.$$

Since $(I - A)^{-1}$ does not exist, the determinant of $(I - A) = 0$. But

$$\text{Det}\,[S(I - A)] = [\text{Det } S]\,[\text{Det } (I - A)] = \text{Det } I.$$

Since Det I = 1 and [Det S] [Det (I - A)] = 0, a contradiction arises. The original assumption was therefore incorrect and the necessity of the existance of $(I - A)^{-1}$ for the convergence of S is proved.

It can now be shown that the matrix A meets the necessary condition for convergence of Theorem 5.

Lemma 2. Given an input matrix X of m, n-dimensional vectors, the rows of the matrix (I - A) are linear combinations of the vectors X_i, where I is the identity matrix and A is the matrix defined by Equation 10.

Proof. As the most simple example, consider the case where X is 2 by n, i.e., there are only two n-dimensional input vectors to be classified. Then

$$I - A = Q_1 + Q_2 - Q_2Q_1.$$

An expression for each row of (I - A) can now be found. R_i represents the i-th row of (I - A), i = 1,2,3,. . . n.

$$R_1 = \left[\frac{x_{11}}{|X_1|^2} - \frac{(X_1 \cdot X_2)\, x_{21}}{|X_2|^2 |X_1|^2}\right] X_1 + \left[\frac{x_{21}}{|X_2|^2}\right] X_2$$

$$R_2 = \left[\frac{x_{12}}{|X_1|^2} - \frac{(X_1 \cdot X_2)\, x_{22}}{|X_2|^2 |X_1|^2}\right] X_1 + \left[\frac{x_{22}}{|X_2|^2}\right] X_2$$

.

.

.

$$R_n = \left[\frac{x_{1n}}{|X_1|^2} - \frac{(X_1 \cdot X_2)\, x_{2n}}{|X_2|^2 |X_1|^2}\right] X_1 + \left[\frac{x_{2n}}{|X_2|^2}\right] X_2.$$

Thus

$$R_1 = a_1 X_1 + b_1 X_2$$

.

.

.

$$R_n = a_n X_1 + b_n X_2$$

where a and b are the coefficients of X_1 and X_2 respectively.

In the general case of m input vectors for which X is an m by n matrix,

$$R_1 = a_1 X_1 + b_1 X_2 + \ldots + q_m X_m$$

.

.

.

$$R_n = a_n X_1 + b_n X_2 + \ldots + q_m X_m.$$

The n rows of the matrix $(I - A)$ are thus linear combinations of the m rows of X and Lemma 2 is proved.

Theorem 6. Given an input matrix X of m, n-dimensional vectors, the matrix $(I - A)^{-1}$ exists only if the m vectors span the n-dimensional space.

Proof. The proof is based on the fact that the inverse of $(I - A)$ will exist if and only if the determinant of $(I - A)$ is non-zero. For the determinant to be non-zero the rows of $(I - A)$ must be independent. From Lemma 1 it is seen that the n rows of $(I - A)$ are linear combinations of the m input vectors. Thus only if n of the m input vectors are independent can the rows of $(I - A)$ be independent. If the m vectors span the n-dimensional space, then n of them must be independent. Thus if the input vectors span the space, the rows of $(I - A)$ are independent and $(I - A)^{-1}$ exists. If the m vectors do not span the n-dimensional space, at least one of the n rows of $(I - A)$ is dependent upon the others, and $(I - A)^{-1}$ does not exist.

Corollary 1. If the number of input vectors, m, is less than the number of dimensions, n, the matrix $(I - A)^{-1}$ does not exist.

Proof. The proof is a direct result of Theorem 6. If m is less than n the vectors cannot span the input space and thus $(I - A)^{-1}$ does not exist.

Thus, an input matrix X composed of m, n-dimensional vectors meets the necessary condition for the convergence of A only if $m \geq n$.

Theorem 7. Using the continuous correction algorithm described previously, the learning matrix W, will converge to the desired optimum value, X^{-1}, provided the organizing set X is a complete set of

linearly independent vectors.

Proof. From Equation 12, a general expression for the final value of the learning matrix is

$$W_f = W_{final} = \left[\sum_{t=1}^{\infty} A^t \right] B \quad ,$$

where A and B are matrices defined in Equations 8 and 9. From Theorems 3 and 4 the final W is shown to be,

$$W_f = (I - A)^{-1} B. \tag{13}$$

What remains to be proved is that W_f is actually X^{-1}. Performing elementary operations on Equation 13 gives

$$I = A + B W_f^{-1}. \tag{14}$$

Once again the complete proof is continued only for a specific example, in this case when X is a 2 by 2 matrix, i.e., two input vectors of two dimensions each. A proof for more vectors and dimensions would proceed in an identical fashion, but would involve a great deal more tedious matrix multiplications. Substituting A and B into Equation 14 gives,

$$I = \left[I - \frac{X_1'X_1}{|X_1|^2} - \frac{X_2'X_2}{|X_2|^2} + \frac{X_2'X_2X_1'X_1}{|X_1|^2|X_2|^2} \right]$$

$$+ \left[\frac{X_1'Z_1^*}{|X_1|^2} \quad \frac{X_2'Z_2^*}{|X_2|^2} - \frac{X_2'X_2X_1'Z_1^*}{|X_1|^2|X_2|^2} \right] W_f^{-1}. \tag{15}$$

Equation 15 can be put in the form

$$\frac{X_1'}{|X_1|^2}(X_1 - Z_1 * W_f^{-1}) + \frac{X_2'}{|X_2|^2}(X_2 - Z_2 * W_f^{-1}) + \frac{-X_2' X_2 X_1'}{|X_1|^2 |X_2|^2}(X_1 - Z_1 * W_f^{-1}) = 0 \tag{16}$$

Equation 16 is satisfied provided

$$X_1 = Z_1 * W_f^{-1}$$

and (17)

$$X_2 = Z_2 * W_f^{-1}.$$

Since $Z_1^* = (1 \quad 0)$ and $Z_2^* = (0 \quad 1)$, Equations 17 can be combined to give

$$X = Z * W_f^{-1} = W_f^{-1} \tag{18}$$

since $Z^* = I$. Equation 18 can be easily converted to

$$W_f = X^{-1} \tag{19}$$

thus completing the proof of Theorem 7.

To demonstrate the fact that $(I - A)^{-1} B$ is in fact equal to X^{-1}, the example of Table 1 is worked out below. From previous calculations,

$$A = \begin{vmatrix} 0 & -1/2 & 0 \\ 0 & 1/4 & -1/2 \\ 0 & -1/4 & 1/2 \end{vmatrix} \quad \text{and} \quad B = \begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix}$$

when

$$X = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{vmatrix} \quad \text{and} \quad Z^* = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}.$$

Then

$$W_f = (I-A)^{-1} B = \begin{vmatrix} 1 & -1 & 1 \\ 0 & 2 & -2 \\ 0 & -1 & 3 \end{vmatrix} \begin{vmatrix} 1/2 & 1/2 & 0 \\ -1/4 & 1/4 & 1/2 \\ 1/4 & -1/4 & 1/2 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & 1 \end{vmatrix} = X^{-1}.$$

An interesting secondary result can be observed from Theorem 6 and Corollary 1. It is shown there that if the number of input vectors, m, is less than the number of dimensions, n, the infinite matrix series given in Equation 12 will not converge. Yet many example problems with m less than n were carried out iteration by iteration. These indicated that the training procedure using the continuous correction algorithm did in fact cause the entire weighting matrix W to converge. This result can be stated as a theorem and proved.

Theorem 8. Given an input matrix X of m linearly independent n-dimensional vectors, where $m < n$, the infinite matrix series

$$W = B + A B + A^2 B + \ldots = \left[\sum_{t=0}^{\infty} A^t \right] B$$

converges to a matrix W_f such that

$$X W_f = I.$$

This is true even though the infinite series $\sum_{t=0}^{\infty} A^t$ taken alone diverges.

Proof. The proof is long and tedious and is presented in detail in the Appendix along with an example problem.

This is a rather unexpected result; that a divergent matrix series multiplied by a constant matrix should converge. This happens because the elements of the matrix A which diverge are cancelled when postmultiplied by the matrix B.

Theorems have been proved concerning the learning matrix and the continuous correction algorithm when the input vectors are linearly independent; both the case in which they span the input space and when they don't. Remaining is the situation in which the matrix attempts to classify a set of dependent vectors. That is, when m, the number of input vectors is greater than n. In most practical problems this will probably be the actual circumstances. From Theorems 5 and 6 it is known that the learning matrix can still converge. However the input matrix X is no longer square so that no optimum matrix X^{-1} can be obtained. Unlike the case where $m < n$, if $m > n$, an optimum matrix W so that $X W = Z^*$ cannot exist. This is true because with n dimensions in each column of W there are only n degrees of freedom in the constraint $X W = Z^*$. But there are m equations to be solved since X is m by n, W is n by k, and Z^* is m by k. When $m < n$, infinite solutions exist; when $m = n$, a single solution, X^{-1}, exists; and if $m > n$, no solution exists. The major usefulness of the continuous correction algorithm is therefore limited to those problems in which $m \leq n$, where convergence to a desired matrix has been proved.

The Rate of Convergence Problem

Of course the training routine for actual pattern recognition problems could not proceed through sufficient iterations (theoretically infinite) for the learning matrix to equal the desired optimum value. The question then arises, "How close to the optimum weighting matrix, W_f, does the learning matrix come after a certain number of iterations?" In general this is a very difficult question and the answer is dependent upon the manner in which vectors are organized into classes, and the order of the vectors in the training set. In the simple case in which the input vectors are orthogonal, the matrix converges to the optimum X^{-1} after one iteration. As an example of the complexity of a general analysis, the mathematical form of the output error matrix after one iteration is derived for the simple three vector case. The desired output is considered to be the identity matrix, i.e., $Z^* = I$.

The actual output after one iteration is given by

$$Z(1) = X\,W(1) = X\,B$$

where B is as defined in Equation 8. The output error is then given by

$$E(1) = Z^* - Z = I - Z = I - X\,B.$$

So that there is sufficient space to write E(1), X_1 is abbreviated "1", X_2 is "2" etc.

$$E(1)=\begin{vmatrix} \frac{(1\cdot)^2}{|1|^2|2|^2}+\frac{(1\cdot3)^2}{|1|^2|3|^2}-\frac{(1\cdot3)(1\cdot2)(2\cdot3)}{|1|^2|2|^2|3|^2} & \frac{(1\cdot3)(2\cdot3)}{|2|^2|3|^2}\frac{1\cdot2}{|2|^2} & \frac{1\cdot3}{|3|^2} \\ \frac{(2\cdot3)(1\cdot3)}{|1|^2|3|^2}-\frac{(1\cdot2)(2\cdot3)^2}{|1|^2|2|^2|3|^2} & \frac{(3\cdot2)^2}{|3|^2|2|^2} & \frac{2\cdot3}{|3|^2} \\ 0 & 0 & 0 \end{vmatrix} \qquad (20)$$

The error after just one iteration is thus a complex function of the inner products of the various input vectors. If the input vectors are orthogonal the inner products are zero and as can be seen from Equation 20 the error matrix is zero. The last row of the error matrix is always zero since the learning matrix always correctly identifies the last vector to which it was adapted. The computation of the error matrix after the second iteration requires the summation of 49 separate 3 x 3 matrices. Thus further analysis along these lines becomes rapidly insurmountable without computer techniques.

It would be useful to have some theory concerning the rate of convergence of the infinite matrix series such as that which exists for scalar series. The convergence is dependent upon the magnitude of the eigenvalues of A, all of which were proved in Theorem 3 to be less than one. The development given by Varga (19) would appear to be useful and the following discussion and theorems are based on material presented there.

Definition 1. Let A be an n by n complex matrix with eigenvalues λ_i, $1 \leq i \leq n$. Then

$\rho(A) = \max \left| \lambda_i \right|$ is the spectral radius of A.

Definition 2. If A is an n by n complex matrix, then

$$\left\| A \right\| = [\rho(A'A)]^{\frac{1}{2}}$$

is the spectral norm of A. That is, the spectral norm is the square root of the maximum eigenvalue of A'A.

The significance of these definitions is given by Theorem 2 which states that a necessary and sufficient condition for an infinite matrix series to converge is that the eigenvalue of A with the maximum absolute value be less than one. The usefulness of this theorem is limited by the difficulty of finding the eigenvalues of higher order matrices.

Definition 3. Let A and B be two n by n complex matrices. If for some positive t, $\left\| A^t \right\| < 1$ then

$$R(A^t) = -\ln \left[\left(\left\| A^t \right\| \right) \right]^{1/t} = \frac{-\ln \left\| A^t \right\|}{t}$$

is the average rate of convergence for m iterations of the matrix A.

If $R(A^t) < R(B^t)$ then B is iteratively faster for t iterations than A. Thus $R(A^t)$ theoretically provides a measure of rate of convergence. Again the problem is the practical impossibility of computing $R(A^t)$ for any matrix A of interest. A simplification is provided for a sufficiently large number of iterations by the following theorem.

Theorem 9. Let A be a convergent n by n complex matrix. For all t sufficiently large the asymptotic rate of convergence, $R_\infty(A)$, is

defined by

$$R_{\infty}(A) = \lim_{t \to \infty} R(A^t) = -\ln \rho(A).$$

Thus the necessary computation is reduced to "merely" finding $\rho(A)$. A further complication is involved however since how large a "t" is "sufficiently large" for the theorem to hold varies for each specific example. The usefulness of the preceding theory is that an expression for the decay rate of the norm of the error matrix for an iterative process can be developed. As mentioned previously, the link between the theoretical development and practical utilization for this method remains to be found.

Discussion

Since the learning matrix only approaches the desired value in the limit, and then only if the organizing set is composed of independent vectors, an output, Z_i, of binary ones and zeros is never going to be obtained. Some means must therefore be used to determine from the actual row vector output Z_i the classification of the input vectors. Two methods might be used. These methods and their relationship to various other pattern recognition techniques are discussed below.

First, a threshold, d, might be placed on each column output. Thus the output of the column would obey the following condition for each input X_i and column weighting vector W_j.

$$z_{ij} = \begin{cases} 1 & \text{if } X_i W_j \geq d_j \\ 0 & \text{if } X_i W_j < d_j \end{cases} \qquad \text{for all } j$$

A perfect classification of the input vectors would result if one and only one column produced a 1 output for each input X_i. The classification reduced to deciding whether an input is or is not a member of the j-th class. The situation is more readily explained in terms of threshold functions.[1] It requires the existance of a hyperplane given by W_j and d_j, such that all vectors which are members of set c_j lie on one side of the hyperplane, and all other vectors lie on the other side. Figure 6a illustrates an example in two dimensional space where the input vectors are separable into three classes by one hyperplane per class.

The present learning routine however has no provision for the computation of the thresholds d_j. A different method of classification is more feasible. This method might be called "peak detection". The values of the output columns, z_{ij}, are scanned and the input is classified into the class c_j corresponding to the maximum column output. Thus, $X_i \in c_j$ if

$$z_{ij} = X_i W_j > X_i W_k = z_{ik} \qquad k \neq j, \text{ for all } k.$$

This method is essentially analogous to classification by correlation coefficients as described in Chapter II. The expression $X_i W_j > X_i W_k$ or

$$X_i(W_j - W_k) > 0 \tag{21}$$

is the equation of a hyperplane passing through the origin with a normal direction given by the vector $(W_j - W_k)$. Thus the various classes may be separated by the second method provided each pair of classes may be sepa-

---

[1] A thorough discussion of threshold functions is given in Winder (20), and (21).

rated by a hyperplane through the origin. This is a stringent condition on the classes but not as strong as that required in the first method for separation. For example, the classes shown in Figure 6b could be separated in this way but not by the threshold method. However, a serious disadvantage of the learning matrix training routine using the continuous correction algorithm is that there is no guarantee the learning matrix will converge to hyperplanes which perfectly separate the classes even if such hyperplanes exist. This difficulty is overcome by a different training algorithm for the learning matrix.

The Error Correction Algorithm

As discussed previously, the learning matrix cannot converge to the desired output Z^* when the vectors of the organizing set are dependent. The continuous correction algorithm attempts to find weighting vectors, W_j, $j \neq i$, which are orthogonal to each input $X_i \in c_i$ so that $X_i W_j = 0$. For a dependent organizing set such weighting vectors don't all exist. If peak detection is used to determine the output class, the learning matrix is adapted even when the output is correct in a peak detection sense. This is because the continuous correction algorithm considers the system in error if the column outputs are not binary in form; that is, a one in the correct column and a zero in all other columns.

A different training routine, hereafter called the error correction algorithm, may be used which adapts the learning matrix only when an error is made in a peak detection sense. This algorithm is essentially the same as the error correction training routine used with the perceptron proposed by Rosenblatt (15). Thus the matrix is altered if and only if for an

input X_i from the c_j th class,

$$z_{ij} \leq z_{ik} \quad \text{for any } k \neq j. \tag{22}$$

The method of adaption is as follows:

$$W_j(t) = \begin{cases} W_j(t-1) & \text{if } X_i W_j(t-1) > X_i W_k(t-1) \text{ all } k \neq j \\ W_j(t-1) + pX_i' & \text{if } X_i W_j(t-1) \leq X_i W_k(t-1) \text{ any } k \neq j \end{cases}$$

$$W_k(t) = \begin{cases} W_k(t-1) & \text{if } X_i W_j(t-1) > X_i W_k(t-1) \text{ all } k \neq j \\ W_k(t-1) - pX_i' & \text{if } X_i W_j(t-1) \leq X_i W_k(t-1) \text{ any } k \neq j \end{cases} \tag{23}$$

where $0 < p \leq 1$. The factor p is used so that only a percentage of X_i may sometimes be added or subtracted to appropriate weight vectors. The reason for this is based on physical considerations in building a learning matrix. For some methods of realization it is necessary to keep the absolute value of each component of the weighting vectors, w_{ij}, which represent the connections between the input and output of the matrix, less than one. For instance, the maximum coefficient of coupling possible between induction coils is unity.

A big incentive for using the error correction algorithm is that it can be proved that if a solution to the pattern classification problem exists in terms of hyperplanes through the origin separating each pair of classes, the learning matrix will converge to such hyperplanes in a finite number of adaptions. The proof follows closely that given by Novikoff (12)

for a similar convergence problem.

The difference between the learning matrix and other pattern recognition techniques (Highleyman (9)) which are based on hyperplane separation, is that the technique presented here finds a weighting vector for each class and a hyperplane for each pair of classes by subtracting the weighting vectors for that pair. In methods that use a separate hyperplane for each pair of the k output classes the necessary number of hyperplanes is given in Chapter II as $k(k - 1) / 2$. With the learning matrix, only k weighting vectors need be found. The drawback however, is that the hyperplane defined by subtracting the weighting vectors of the learning matrix are constrained to pass through the origin of the n-dimensional space whereas the class-pair hyperplanes defined by Highleyman (9) may be located anywhere in space. Thus pattern recognition problems exist in which perfect classification is possible with $k(k - 1) / 2$ hyperplanes but for which the learning matrix cannot provide a perfect solution. This ability is sacrificed for simplicity in that less weighting vectors must be found.

To prove the theorem concerning the convergence of the error correction algorithm, a problem is considered in which a group of vectors in an n-dimensional space are to be classified into two output sets. An extension of the proof to more than two sets can easily be made by considering each two sets separately. The two sets of inputs are designated $A = (\alpha_1, \alpha_2, \ldots \alpha_g)$ and $B = (\beta_1, \beta_2, \ldots \beta_h)$. The method of classification is by peak detection. That is, two weighting vectors, W_a and W_b, are to be found such that the following conditions are met:

$$\alpha_i \cdot W_a > \alpha_i \cdot W_b + d \qquad \text{for } i = 1,2, \ldots g$$

$$\beta_j \cdot W_b > \beta_j \cdot W_a + d \qquad \text{for } j = 1,2, \ldots h.$$

The constant d is the discrimination level. It is a sort of safety factor to ensure the classes are sufficiently separated. In many cases it may be set equal to zero. Figure 10 shows a two dimensional space where the sets A and B are designated. Possible weighting vectors W_a and W_b are also shown and the significance of the discrimination level d can be seen. The sets are separated by parallel hyperplanes each a distance d from the origin. Thus for perfect classification of the inputs, the sets must be a distance 2d apart.

The method for obtaining suitable weighting vectors is the error correction algorithm of training the learning matrix as given in Equation 23. A further explanation is given as follows: Construct an input sequence, S, of vectors taken from sets A and B. One method would be to alternate between sets A and B although this is not necessary. Thus an input vector is selected and dotted into W_a and W_b. If the wrong response results (in a peak detection sense) the weighting vectors are adapted by adding the input vector to the weighting vector corresponding to the correct classification and subtracting the input vector from the other weighting vectors in accordance with Equation 23.

The next theorem pertains to the convergence of the error correction algorithm.

Theorem 10. Consider two sets of n-dimensional vectors, A = $(\alpha_1, \alpha_2, \ldots \alpha_g)$ and B = $(\beta_1, \beta_2, \ldots \beta_h)$. If the pair of sets

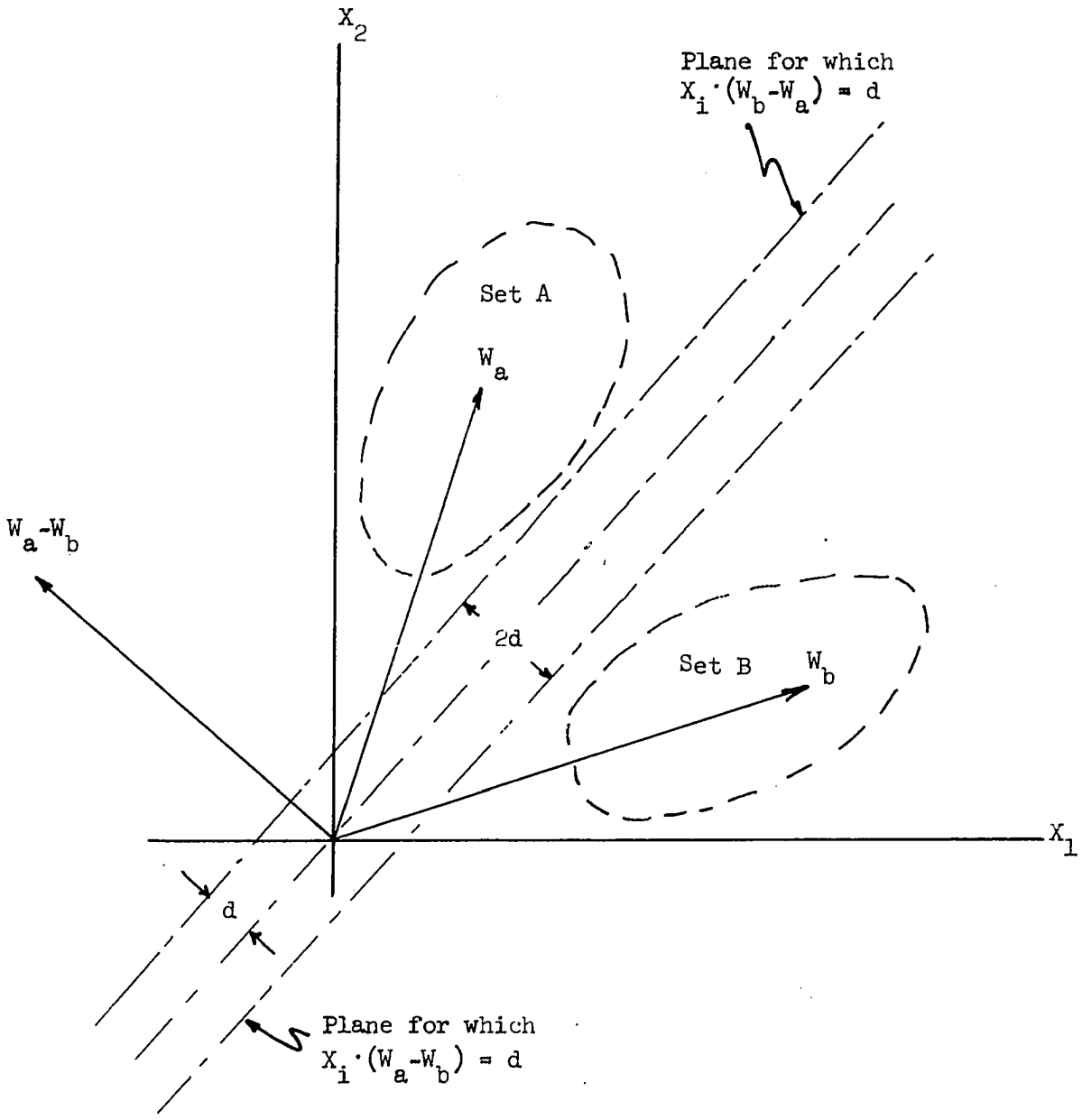


Figure 10. Separating hyperplanes with discrimination level d

is separable by a hyperplane through the origin with normal direction given by $W_a^*-W_b^*$, that is, if <u>any</u> weighting vectors W_a^* and W_b^* exist so that all inputs can be properly identified in a peak detection sense, the error correction algorithm will converge to <u>some</u> vectors, W_a and W_b, which will perfectly classify the inputs of A and B.

<u>Proof</u>. It is known from the hypothesis of the theorem that some vectors W_a^* and W_b^* exist such that

$$\alpha_i \cdot W_a^* > \alpha_i \cdot W_b^* + d \tag{24}$$

and

$$\beta_j \cdot W_b^* > \beta_j \cdot W_a^* + d. \tag{25}$$

These can be rearranged to give

$$\alpha_i \cdot (W_a^* - W_b^*) > d \tag{26}$$

$$\beta_j \cdot (W_b^* - W_a^*) > d. \tag{27}$$

By letting $W_a^* - W_b^* = V^*$ and $\beta_j = -\beta_j$, Equations 26 and 27 are given by

$$\alpha_i \cdot V^* > d \tag{28}$$

$$\beta_j \cdot V^* > d. \tag{29}$$

Now since all inputs, α_i and β_j, satisfy the condition as given in

Equations 28 and 29, each input can be designated as simply X_i. The condition that perfect classification of all inputs is possible then is given by

$$X_i \cdot V^* > d. \tag{30}$$

Equation 30 is true for all inputs α_i and β_j.

A new input sequence is now constructed composed only of those inputs for which an adaption to the weighting vectors occurs. Since no change is made in the weighting vectors for inputs classified correctly, these inputs could be ignored. A training sequence is then left for which adaption takes place with every input. The scalar "r" will be used to indicate the r-th input of the training sequence. Since correction takes place with every input, for each r, the equation below must hold.

$$X(r) \cdot V(r) < d \tag{31}$$

where $V(r) = W_a(r) - W_b(r)$. If Equation 31 were not true, no adaption would take place. But these inputs have been removed by definition.

The assertion of Theorem 10 is that r, which counts the number of adaptions, is finite. Thus after some point, all inputs are classified correctly. An equivalent statement is that Equation 30, which says all inputs are classified correctly for some V^*, and Equation 31, which states an adaption occurs for the present $V(r)$, cannot continue to hold for all r. Eventually $V(r)$ will come sufficiently close to the ideal V^* so that all inputs are classified correctly.

From the error correction algorithm with p = 1, it can be shown

$$V(r) = W_a(r - 1) - W_b(r - 1) + \Delta V$$

$$V(r) = [W_a(r - 1) + X(r)] - [W_b(r - 1) - X(r)]$$

$$V(r) = V(r - 1) + 2\,X(r). \tag{32}$$

Thus

$$V(r) = V(0) + 2\,[X(1) + X(2) + \ldots + X(r)]\,.$$

Taking the dot product of V* and V(r) gives, as a function of r,

$$V(r)\cdot V^* = V(0)\cdot V^* + 2\,X(1)\cdot V^* + \ldots + 2\,X(r)\cdot V^*.$$

Now using Equation 30,

$$V(r)\cdot V^* \geq V(0)\cdot V^* + 2\,r\,d. \tag{33}$$

From Schwartz' inequality it is known that

$$\left| V(r)\cdot V^* \right|^2 \leq \left| V(r) \right|^2 \left| V^* \right|^2 \tag{34}$$

Thus using 33 and 34,

$$\left| V(r) \right|^2 \geq \frac{[V(0)\cdot V^* + 2\,r\,d]^2}{\left| V^* \right|^2}\,. \tag{35}$$

Thus a <u>lower</u> bound for V(r) has been established.

Now using 31 and 32 it follows that

$$\left|V(r)\right|^2 - \left|V(r-1)\right|^2 = \left|V(r-1) + 2\,X(r)\right|^2 - \left|V(r-1)\right|^2$$

$$= 4\,V(r-1)\cdot X(r) + 4\left|X(r)\right|^2$$

$$\leq 4\,(d + M) \tag{36}$$

where $M \geq \left|X(r)\right|^2$. Summing both sides of Equation 36 as r ranges from r = 1,2, . . . gives,

$$\sum_{k=1}^{r} \left[\, V(k)^2 - V(k-1)^2 \,\right] \leq \sum_{k=1}^{r} 4\,(d + M)$$

$$\left|V(1)\right|^2 - \left|V(0)\right|^2 + \left|V(2)\right|^2 - \left|V(1)\right|^2 + \ldots \leq 4r\,(d + M)$$

$$\left|V(r)\right|^2 - \left|V(0)\right|^2 \leq 4r\,(d + M)$$

$$\left|V(r)\right|^2 \leq \left|V(0)\right|^2 + 4r\,(d + M). \tag{37}$$

Thus an <u>upper</u> bound for V(r) has also been derived. Equations 35 and 37 cannot continue to hold for all r. Thus some point is reached where adaptions no longer take place and Theorem 8 is proved.

An upper bound for r can be found by substituting Equation 35 into 37. For simplification, V(0) is assumed to be 0. Then

$$r \leq \frac{4\,(d + M)\left|V^*\right|^2}{4\,d^2}. \tag{38}$$

Note that as d becomes small, the upper bound on r becomes greater. This occurs since d is a measure of the amount of separation between the classes. The closer the classes are, the more adaptions required for perfect classification. The separation distance d cannot be zero since then the hypothesis that the classes are in fact separable would not hold.

Actual values for r were found by Ishida (10) in a computer simulation of threshold elements. His study is analogous to the separation of inputs into two classes as presented here. A comparison of the theoretical upper bound given by Equation 38 and the actual adaptions required for perfect separation of two classes as presented by Ishida discloses that the average computed value is roughly 20 percent of the upper bound. The upper bound is thus quite conservative.

An example problem is given in Tables 2 and 3 in which four inputs are classified into two classes using each of the training algorithms. The two output sets for these tables are $S_1 = \{(1\ 0\ 0),\ (1\ 0\ 1)\}$ and $S_2 = \{(1\ 1\ 1),\ (0\ 1\ 1)\}$. The other necessary data are,

$$X = \begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}, \qquad W(0) = \begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix} \qquad \text{and} \qquad Z^* = \begin{vmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{vmatrix}.$$

Table 2. Two iterations of training the learning matrix using the continuous correction algorithm to classify four inputs into two output sets.

Iteration t	Matrix	Input X_1	X_2	X_3	X_4
	Z	(0 0)	(1 0)	(1/3 2/3)	(-1/3 1/3)
1	ΔW	$\begin{vmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$	$\begin{vmatrix} -1/3 & 1/3 \\ -1/3 & 1/3 \\ -1/3 & 1/3 \end{vmatrix}$	$\begin{vmatrix} 1/3 & -1/3 \\ 0 & 0 \\ 1/3 & -1/3 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 \\ 1/6 & 1/3 \\ 1/6 & 1/3 \end{vmatrix}$
	W	$\begin{vmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 2/3 & 1/3 \\ -1/3 & 1/3 \\ -1/3 & 1/3 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 \\ -1/3 & 1/3 \\ 0 & 0 \end{vmatrix}$	$\begin{vmatrix} 1 & 0 \\ -1/6 & 2/3 \\ 1/6 & 1/3 \end{vmatrix}$
	Z	(1 0)	(1 1)	(1/2 1/3)	(-5/12 5/6)
2	ΔW	$\begin{vmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$	$\begin{vmatrix} -1/3 & 0 \\ -1/3 & 0 \\ -1/3 & 0 \end{vmatrix}$	$\begin{vmatrix} 1/4 & -1/6 \\ 0 & 0 \\ 1/4 & -1/6 \end{vmatrix}$	$\begin{vmatrix} 0 & 0 \\ 5/24 & 1/12 \\ 5/24 & 1/12 \end{vmatrix}$
	W	$\begin{vmatrix} 1 & 0 \\ -1/6 & 2/3 \\ 1/6 & 1/3 \end{vmatrix}$	$\begin{vmatrix} 2/3 & 0 \\ -1/2 & 2/3 \\ -1/6 & 1/3 \end{vmatrix}$	$\begin{vmatrix} 11/12 & -1/6 \\ -1/2 & 2/3 \\ 1/12 & 1/6 \end{vmatrix}$	$\begin{vmatrix} 11/12 & -1/6 \\ -7/24 & 3/4 \\ 7/24 & 1/4 \end{vmatrix}$

The output matrix after two iterations is thus given by,

$$Z = X\,W(4,2) = \begin{vmatrix} 11/12 & -1/6 \\ 11/12 & 5/6 \\ 29/24 & 1/12 \\ 0 & 1 \end{vmatrix}.$$

The only misclassified input is X_2 which is placed in S_1 rather than S_2 since 11/12 > 5/6.

Use of the error correction algorithm on the same problem is given in Table 3.

Table 3. Classification of four inputs into two output sets using the error correction algorithm.

Iteration t	Matrix	Input X_1	Input X_2	Input X_3	Input X_4
1	Z	(0 0)	(1 -1)	(-1 1)	(-1 1)
	ΔW	1 -1 0 0 0 0	-1 1 -1 1 -1 1	1 -1 0 0 1 -1	0 0 0 0 0 0
	W	1 -1 0 0 0 0	0 0 -1 1 -1 1	1 -1 -1 1 0 0	1 -1 -1 1 0 0
2	Z	(1 -1)	(0 0)	(-1 1)	(-2 2)
	ΔW	0 0 0 0 0 0	-1 1 -1 1 -1 1	1 -1 0 0 1 -1	0 0 0 0 0 0
	W	0 0 0 0 0 0	0 0 -2 2 -1 1	1 -1 -2 2 0 0	1 -1 -2 2 0 0

The output matrix after two iterations with this algorithm is

$$\therefore X W = Z = \begin{vmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -2 & 2 \end{vmatrix}.$$

All inputs are classified correctly and no further adaptions ever need be made to the learning matrix. The scalar r in this example is five, since five adaptions were made during the training process.

The hyperplane which divides the two classes can be determined for each algorithm by subtracting the weighting vectors. The resulting vector is the normal to a plane passing through the origin. For the continuous correction algorithm,

$$V_{12} = W_1 - W_2 = (11/12 \quad -7/24 \quad 7/24) - (-1/6 \quad 3/4 \quad 1/4)$$

$$= (13/12 \quad -25/24 \quad 1/24).$$

For the error correction algorithm,

$$V_{12} = W_1 - W_2 = (1 \quad -2 \quad 0) - (-1 \quad 2 \quad 0)$$

$$= (2 \quad -4 \quad 0).$$

The first plane misclassifies one input, X_2, while the second plane perfectly separates the two sets. The error correction algorithm is therefore the superior training method.

Two topics remain to be discussed. These are analog inputs, and the generalizing ability of the learning matrix.

Analog Inputs

Analog inputs, cases in which the values of the input vectors, x_1, x_2, . . ., x_n may be any value, give the learning matrix no particular difficulty. The length of the input vectors using the binary variables 0 and 1 varies anyway. If binary variables 1 and -1 are used, all inputs have a length $[n]^{1/2}$ where n is the number of dimensions. Using the continuous correction algorithm, normalization takes place with each adaption since $W = X_i'E_i / |X_i|^2$. Thus each component of ΔW is divided by a normalization factor $|X_i|^2$. A factor of normalization could also be employed with the error correction algorithm, either by varying the parameter p in Equations 23, or by dividing each component of X_i by $|X_i|^2$.

The classification performed by the learning matrix is changed if each of the weighting vectors is normalized after the training period is terminated. With no normalization, two classes are separated by a hyperplane through the origin defined by $H \cdot W_1 = H \cdot W_2$ where H is a vector lying in the hyperplane. This hyperplane has a normal direction given by $W_1 - W_2$. If each of the weighting vectors is normalized, so that $\sum_{i=1}^{n} w_{ij}^2 = 1$, all weighting vectors have the same length and can be considered to terminate on a hypersphere. Classification is then by angular similarity entirely. An input is associated with the weighting vector with which it forms the smallest angle. In many problems, classification by normalized weighting vectors will be an improvement over the non-normalized weighting matrix. In addition, the normalization forces all weighting matrix values, w_{ij}, to be less than one, a desirable condition for some methods of physical implementation.

Generalizing Ability

The question of the generalizing ability of the pattern recognition system may be phrased as follows: "How well will the learning matrix separate patterns which are not included in the training set but which are to be separated into the same classes?" This is a question which involves the error rate on new patterns and can be answered only by testing the new data. It is reasonable to assume that new patterns which are close to old patterns will be similarly classified. The importance of having patterns in the training set which truly represent the pattern class is thus dramatized. If the training set does not provide a true cross section of an input class, new patterns will likely be misclassified. The performance of the learning matrix is demonstrated in another chapter where the results of a computer simulation are presented.

ELEMENTS FOR LEARNING MATRIX IMPLEMENTATION

The physical realization of the learning matrix hinges on the availability of variable weighting elements, w_{ij}, to serve as a connection between the input and output lines of the matrix. Such components must have several distinctive properties.

Ideally the gain or value of these elements should be continuously variable, at least over a normalized range of plus one to minus one. Once set, the value of the gain should be permanent until further adaption takes place. A linear relationship would be desirable between the adaption signal and the gain of the element but such a relationship is not absolutely necessary.

The element also must provide a means for non-destructive readout of its stored value. This readout should be in the form of the product of a sense signal x_i and the stored value, w_{ij}. Thus upon application of the sense signal, x_i, (often a current or voltage) an output $x_i w_{ij}$ is obtained. For example, in the learning matrix, the output of each column, z_{ij}, is $z_{ij} = \sum_{k=1}^{n} x_{ik} w_{kj}$.

The most significant results so far in efforts to develop variable weights for learning networks have been in the area of magnetic phenomenon. One reason for this may be the intensive research effort which magnetics has received as a result of the success of magnetic memories for digital computers. The development of several recent devices which could be used to implement the learning matrix is discussed briefly.

Hawkins (6) has proposed a ferrite toroid which he calls a "magnetic integrator" as a variable weight element. The value of the weight is stored as the amount of flux linking the toroid and this is varied by

means of an adapt winding on the toroid. Because of hysteresis loss in the toroid the residual flux can be set at various levels. Non-destructive readout is obtained by the application of a magnetic field orthogonal to the normal direction of magnetization of the toroid. The change in the magnetization vector when the sense field is applied is picked up by a sense coil on the toroid. The voltage induced is proportional to the amount of stored flux and the magnitude of the sense field. A small adaptive array (five elements) was constructed using this procedure and satisfactory results were reported.

A different approach is reported by Crafts (2). His element consists of a pair of tape wound cores driven from a radio-frequency power source. A sense winding also links the cores - being wound one direction on one core and the opposite direction on the other core. With this arrangement, shown in Figure 11a, the fundamental component of radio-frequency voltage in the sense winding is cancelled but a second-harmonic distortion voltage is left which is proportional to the remanent flux in the cores. The remanent flux level can be altered by passing a direct current through the output winding. Due to an interaction between the dc adaption current and the rf drive current, the time rate of change of the remanent flux, with respect to the adaption current, is quite constant and reversible, thus providing a smoothly variable gain with permanent memory.

The element with perhaps the most promise is an adaptive, magnetic thin film device which is described in Pohm, Allen, and Nilsson (13). This device, shown in Figure 11b, consists of two film-coated wires; one of which is used as a storage element and the other as a sensing element. In the storage element the easy direction of the film is axial while the

sense element has a circumferential easy direction. The sensing element behaves as a magnetic film balanced modulator with an output which is approximately a product of the field which represents the stored weight and the derivative of an input sensing field. Preliminary experimental results on a two by two array indicate satisfactory performance although many engineering problems remain to be solved.

Since the learning matrix can be successfully simulated on a digital computer, the question might be asked, "What can adaptive networks do that computers cannot already do?" The answer is, in most cases, nothing. However, an adaptive network or learning matrix still has many advantages over a digital computer. First of all, adaptive networks, being inherently parallel data processors, cannot be simulated efficiently with a computer which operates in a serial fashion. As pointed out in the next chapter on computer simulation, up to ten minutes of 7074 time was required for simple recognition experiments. Adaptive networks not only operate on all the input vector components in parallel, but also perform operations of multiplication and summation in parallel. Thus a big advantage of an actual adaptive device is its speed of operation.

Adaptive networks could also be designed to accept analog inputs directly from the environment. A digital computer would in general require a greater amount of proprocessing of the input data.

In addition, there are many pattern recognition applications for which the cost of an all purpose digital computer could not be justified. An adaptive network specially designed for a specific group of tasks would be much more economical.

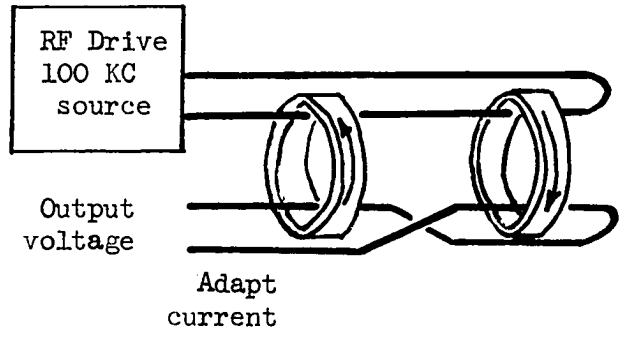


Figure 11a. Second-harmonic variable-gain component of Craft (2)

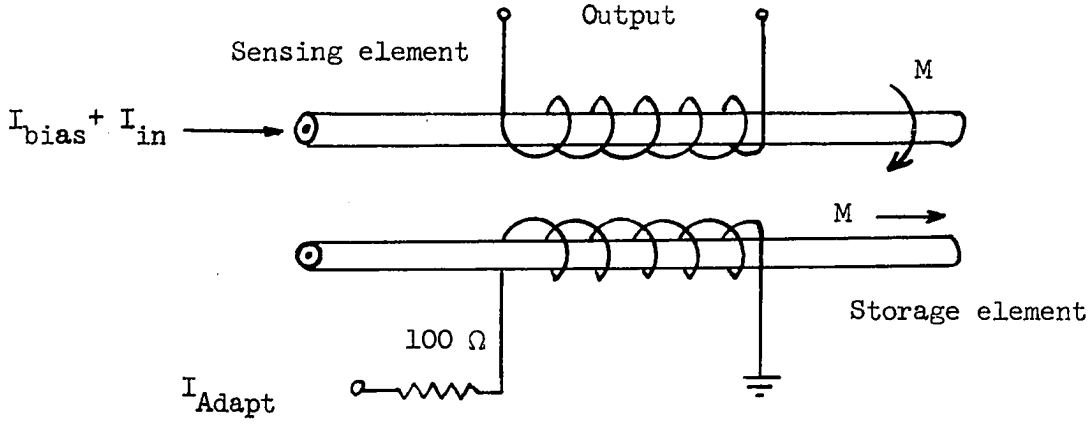


Figure 11b. Adaptive thin-film weighting element of Pohm et al. (13)

The ultimate choice of an adaptive element for learning matrix applications must be based upon the engineering criteria of economy, reliability, and versatility. If suitable components can be developed, the inherent advantages of adaptive systems can be incorporated into future data-processing machines. Such systems could combine the advantage of adaptive networks with the versatility of digital computers to achieve a machine with valuable properties of both systems.

COMPUTER SIMULATION OF THE LEARNING MATRIX

Introduction

The learning matrix was simulated on a digital computer to test the previous theory on a practical pattern recognition problem. As suggested by Rosenblatt (15) a simulation program for pattern recognition should fulfill three basic conditions.

1) Simulation should not be attempted without a theoretical analysis of the pattern recognition system. Such an analysis should suggest suitable experimental parameters and should indicate questions of theoretical interest.

2) Suitable measures of performance must be defined. This means that some task must be set for the system, the outcome of which can be clearly recognized and quantified.

3) Experiments must be designed with suitable controls against trivial or ambiguous results.

The simulation was planned with these three conditions in mind. The mathematical considerations of the previous chapter provide a theoretical background to be tested in the simulation. The measure of performance set for the learning matrix is the percentage of correct classifications of input patterns. The experiments to be described provide a rigorous test of the ability of the matrix to perform a pattern recognition function.

The Iowa State University Computation Center IBM 7074 Digital Computer was used for the simulation. An abbreviated flow diagram for the Fortran program is shown in Figure 12. Actually two programs were developed; one for the continuous correction algorithm and another for the error correction algorithm. These different programs are shown by the

dotted lines in Figure 11.

Description of the Computer Program

The first step in the program is the input of the vector, X_i. In general this could represent any type of pattern; a letter, a number, a waveform, any output from the transducer portion of the pattern recognition system. This vector is postmultiplied by the weighting matrix W to give the output row matrix Z_i. A branching is then made depending upon whether the operation of the system is presently in the training or the recognition mode. If in the training mode, the learning matrix is adapted according to one of the two algorithms described in the previous chapter. Adaption then proceeds with the remainder of the organizing set for a number of iterations fixed by the experiment. If in the recognition mode, the maximum component of Z_i is found (which represents the assigned pattern class) and compared with the true class for this input. A counter for correct or incorrect responses is stepped and the program either proceeds to recognize the next input, or returns to the training mode for additional adaption of the learning matrix. Provision is also made to compute the percentage of correct responses during any experiment.

The performance of any recognition method depends not only upon the system itself, but also upon the type of characters or patterns encountered by the system. The recognition rate also depends strongly upon the amount of preprocessing a pattern receives before being presented to the categorizer. Comparison among various recognition methods is difficult, especially if the methods do not operate on the same data.

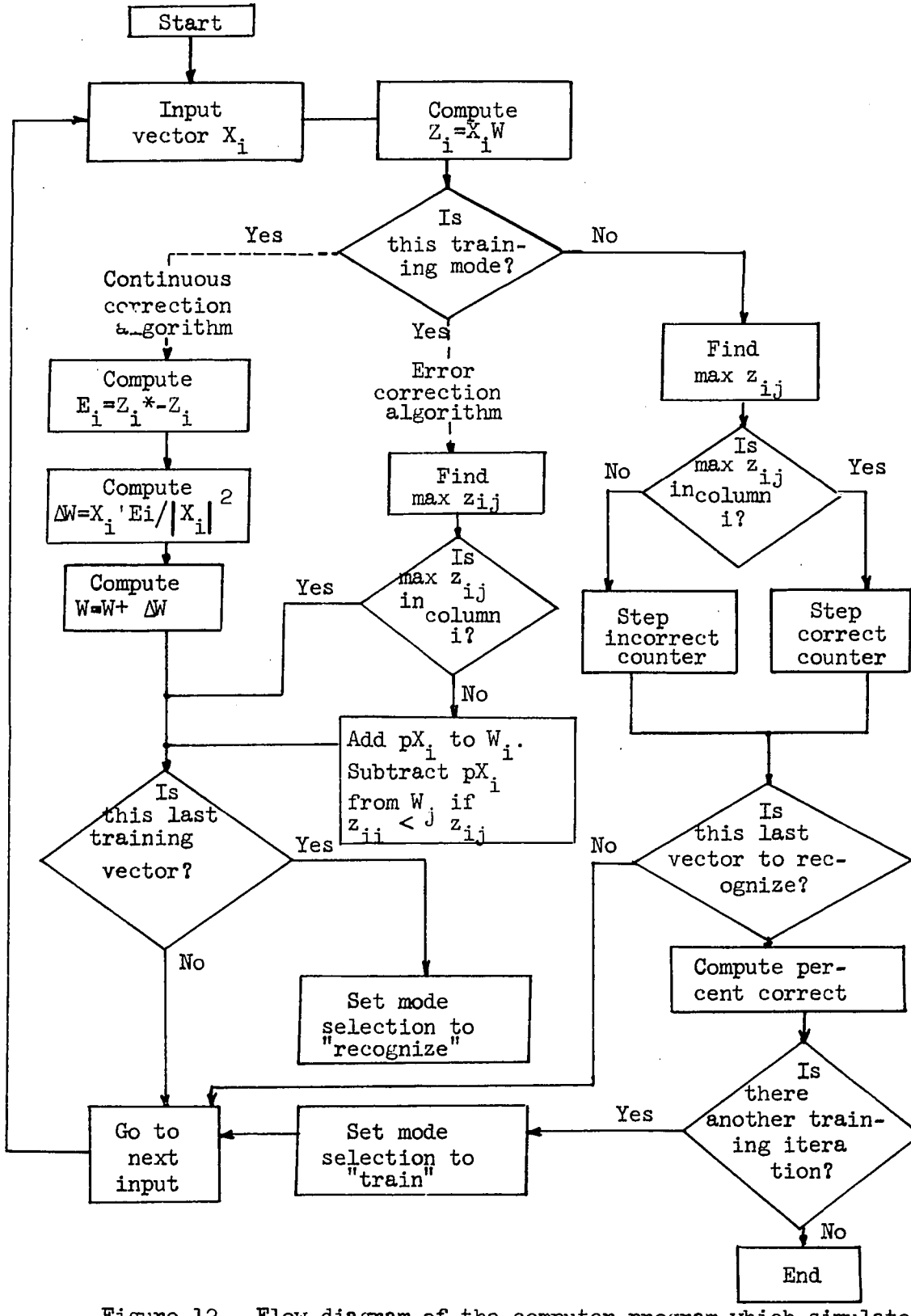
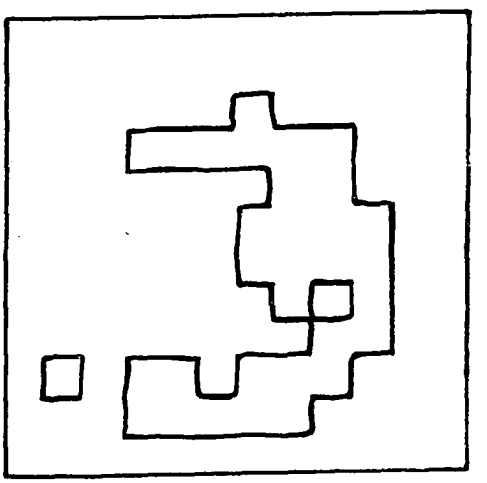


Figure 12. Flow diagram of the computer program which simulates the learning matrix

Description of the Patterns

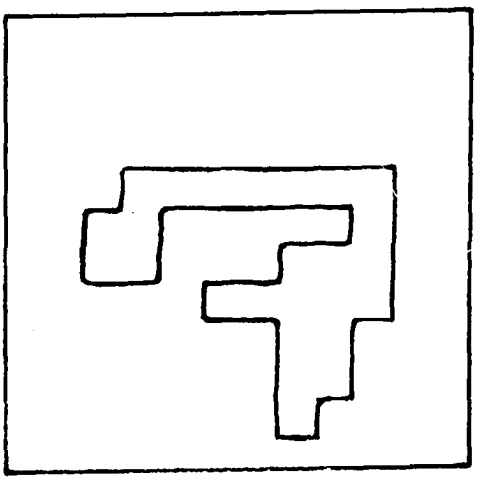
To help establish a reference for comparison, the experiments reported here are based upon a set of hand-printed numbers prepared and used by Dr. W. H. Highleyman (8,9), formerly of the Bell Telephone Laboratories. The data consist of 50 sets of hand-printed arabic numerals, zero through nine, and 50 sets of machine-printed numbers from an IBM 407 line printer. The hand-printed numbers were printed by 50 different people. These persons were required to print neatly on $\frac{1}{4}$-inch quadrilled paper at a size approximating the rules boxes on the paper. Some samples of the data are given in Highleyman (8) and Figure 13. The data were then automatically reduced to a 12 x 12 binary matrix by an optical matrix scanner and encoded on punched cards with an octal code. The use of the octal code on the cards required a subroutine in the simulation program which converted the octal code back to a binary representation. Since the characters are quantified onto a 12 x 12 matrix, each pattern is represented by a 144-dimensional vector, each dimension being a binary one or zero. The characters were roughly centered by using center of gravity alignment. However, the character size was not normalized and the size variation is about two to one. These cards, obtained through Dr. W. H. Highleyman are the input vectors, X_i, to the simulation program. The machine printing is fairly uniform but some of the hand-printed numbers are crude to say the least. To demonstrate how bad some of the handprinting actually was, 50 of the hand-printed numbers were drawn on cards. Ten people were asked to recognize the numbers and the resulting average human recognition rate was only 78%. The numbers thus contain a large amount of "noise" since no two patterns in the same class were exactly alike.

Figure 13. Examples of hand-printed numbers. The negative numbers below each figure represent the highest and next highest column outputs after four iterations using the error correction algorithm. Figures a) and b) are correctly identified; c) and d) are mistakes



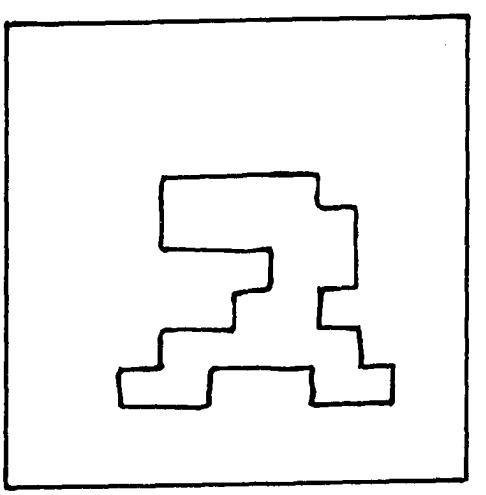
3 -33.1
7 -37.4

a)



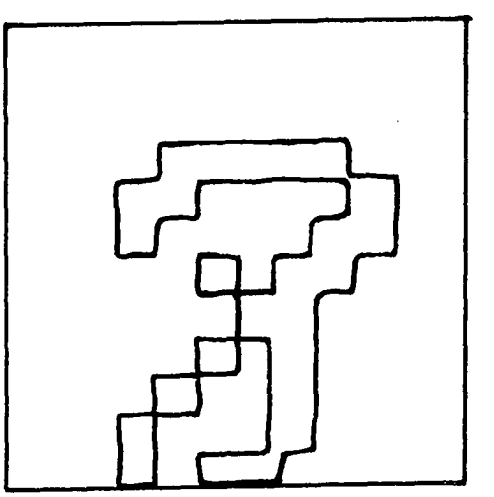
9 -29.7
0 -32.0

b)



5 -32.2
9 -32.5
2 -32.9

c)



3 -26.3
8 -27.0

d)

This points out the difficult job of classification a pattern recognition system has with this set of data.

Actually additional data was available in the form of 50 sets of hand-printed alphabets. However a limited amount of computer time made experiments based on the entire set of alphanumeric data impossible. To complete one iteration using all 50 sets of both letters and numbers would have required approximately six hours of IBM 7074 time. The actual number of experiments which could be performed on just the numerical data were in fact limited by the amount of unsponsored research time which could be obtained for this project. The total time used in all the simulated learning matrix experiments was 100 minutes. Projects of this type require excessive time due to the inherently complex multiplications which must be completed. With 144 dimensions and ten output classes the weighting matrix has 144 rows and ten columns. One computation of $Z = X W$ thus requires $144 \times 10 = 1440$ multiplications.

Results

The experiments were designed to test the recognition rate of the learning matrix using each of the training algorithms. Two types of experiments were performed; one in which the number of vectors in the organizing set was less than the number of dimensions, and the other in which the organizing set was composed of all 500 inputs. Also investigated was the effect of a threshold or discrimination level to provide for rejection of questionable decisions.

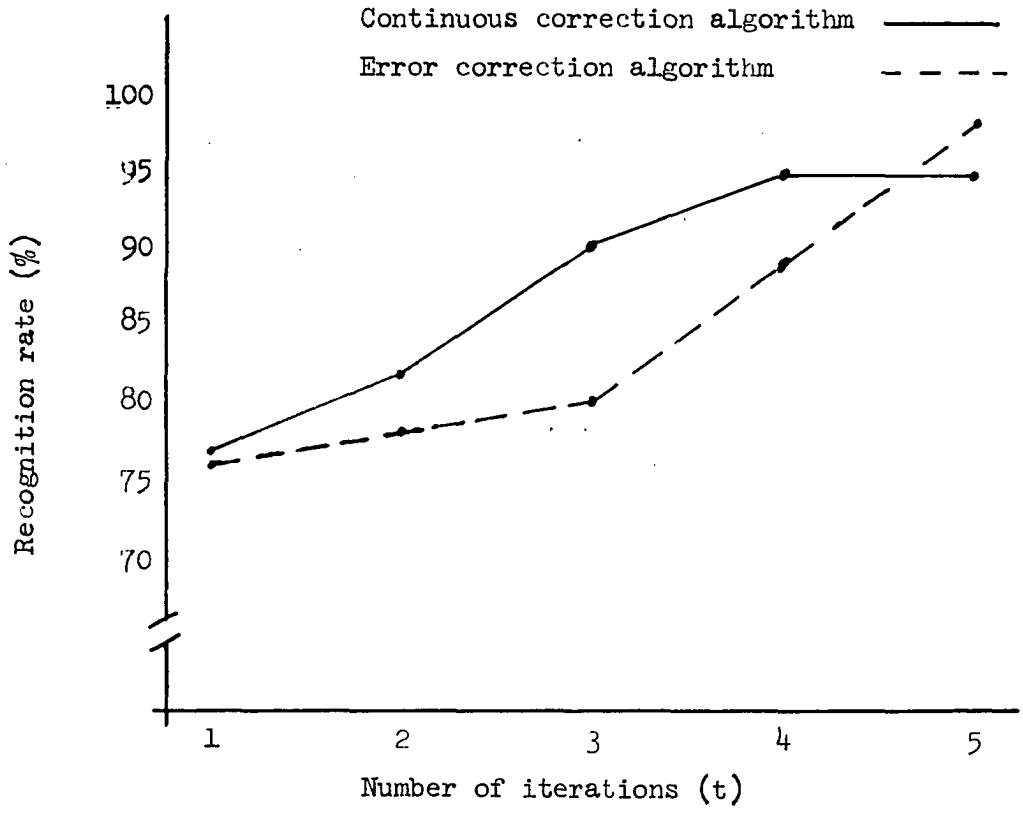
In the first experiment, an organizing set of 100 vectors, (ten ones, ten twos, etc.) was used to adapt the learning matrix. The pattern

classes each were presented in turn - that is, the first one, the first two, and up through the first zero and then the second one etc. After each training iteration with these 100 vectors, 120 vectors were classified by the matrix. The first 100 of these being the organizing set and the next twenty inputs being numbers which the matrix had not previously "seen". Table 4 and Figures 14a and 14b show the recognition rate of the matrix for the machine-printed numbers using the continuous correction procedure and also for the hand-printed numbers using the error correction algorithm.

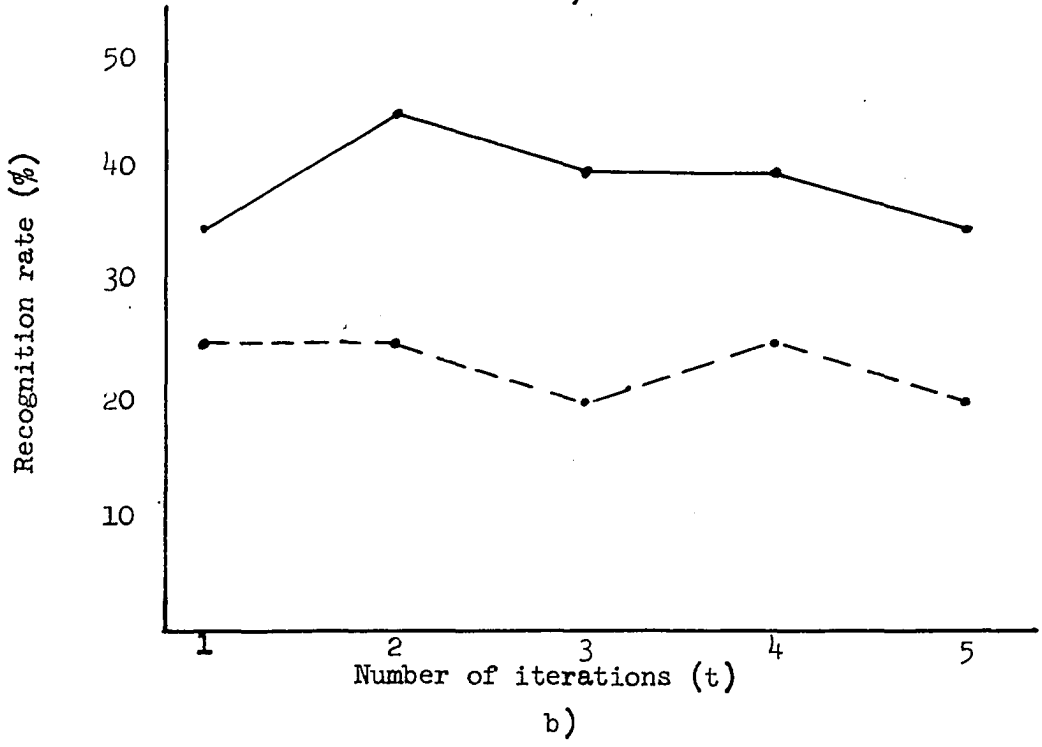
Table 4. Recognition rates for experiments with 100 inputs in the training set.

Type of Data	Algorithm used	Iteration number	Recognition rate (%)	
			Organizing set (100)	New inputs (20)
Machine-printed	Continuous correction	1	99	100
		2	99	100
		3	99	100
		4	99	100
		5	99	100
Hand-printed	Continuous correction	1	77	35
		2	84	45
		3	91	40
		4	95	40
		5	95	35
Hand-printed	Error correction	1	76	25
		2	78	25
		3	80	20
		4	87	25
		5	98	20

Figure 14. Recognition rate versus number of iterations for an organizing set of 100 hand-printed inputs.
a) Recognition rate for the 100 inputs which were members of the organizing set
b) Recognition rate for the 20 inputs which were not members of the organizing set


Continuous correction algorithm
Error correction algorithm
Recognition rate (%)
100
95
90
85
80
75
70
1
2
3
4
5
Number of iterations (t)

a)


Recognition rate (%)
50
40
30
20
10
1
2
3
4
5
Number of iterations (t)

b)

Note that the machine printing was nearly perfectly classified (one error) by the matrix. This indicates the importance of the type of data used in any experiment. Using the continuous correction algorithm, the handprinting recognition rate increased from 77 percent one iteration to 95 percent after four iterations. Although the rate did not increase from four to five iterations, examination of the outputs in each case indicate the size of the error on the five mistakes which were made had decreased. Thus convergence was still continuing. The generalizing ability of the matrix was poor; the rate on the 20 inputs not members of the organizing set being 35-40 percent, which did not improve with further training. This indicates that the organizing set size was not large enough to include the many variations that occur within each pattern class. No further training will improve the recognition rate on these inputs which are insufficiently similar to the patterns in the organizing set for correct recognition to occur.

Using the error correction algorithm, the recognition rate increased to 98 percent after five presentations of the organizing set. The value of p (see Equation 23) used in the algorithm was 0.1. Thus as explained in the previous chapter, ten percent of the input vector was added or subtracted to appropriate weighting matrix columns when an adaption occurred. The value of p which is used makes no difference in the recognition rate. It only affects the size of the weight elements w_{ij}. The weight elements were all negative as shown from the column outputs in Figure 13. This simply means inputs were more often subtracted than added to column vectors during the adaption process. The highest column output is thus the least negative. Once again the recognition rate on inputs not in the

organizing set was poor. Figures 13c and 13d show two inputs which were incorrectly identified with the error algorithm after four iterations. The highest and second highest column outputs are also shown below the figures.

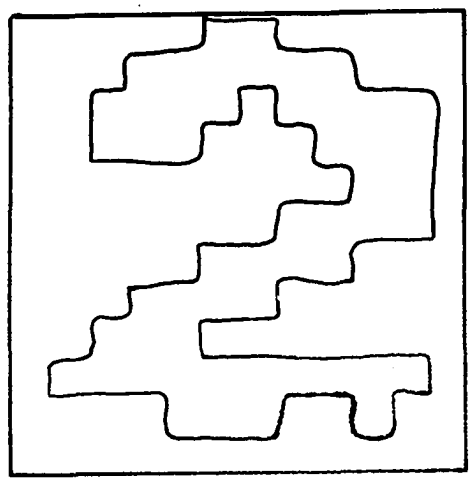
In another experiment, the matrix was trained with each algorithm on all 50 sets of numbers, 500 inputs in all. Table 5 presents these results.

Table 5. Recognition rates for experiments with 500 inputs.

Type of data	Algorithm used	Iteration number	Recognition rate (%)
Machine-printed	Continuous correction	1	99.2
Hand-printed	Continuous correction	1	57.4
		2	60.0
Hand-printed	Error correction	1	64.6

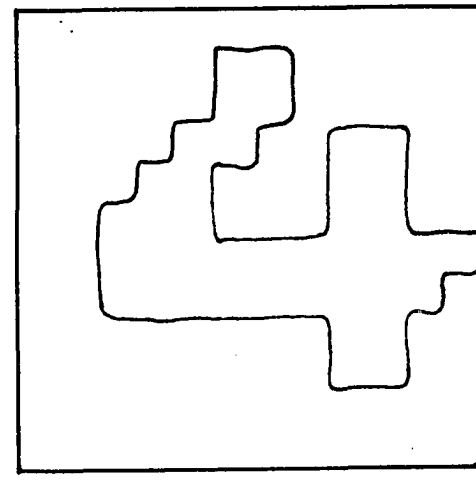
Once again the matrix classified more than 99 percent of the machine-printed numbers correctly. Figures 15c and 15d show two machine-printed numbers upon which mistakes were made. Using the continuous correction algorithm with the hand printing, the recognition rate increased from 57.4 percent after one iteration to 60.0 percent after two iterations. The error correction algorithm classified 64.6 percent of the 500 inputs correctly after only one iteration.

Figure 15. Examples of machine-printed numbers. The numbers below each figure represent the highest and second highest column outputs after one iteration with 500 inputs using the continuous correction algorithm. Figures a) and b) are identified correctly; c) and d) are mistakes



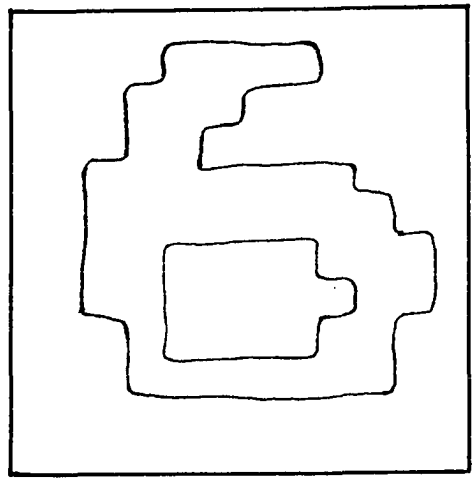
2 0.914
9 0.165

a)



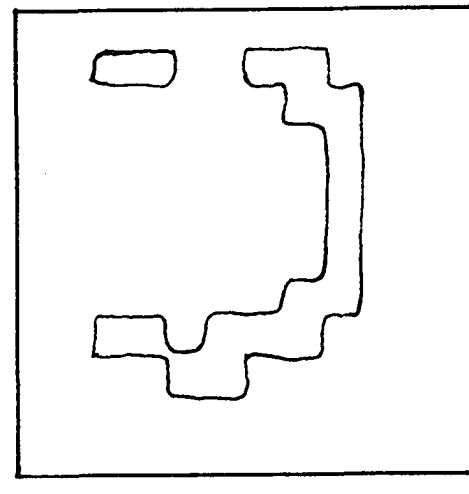
4 0.963
6 0.160

b)



8 0.613
6 0.233

c)



3 0.355
0 0.333

d)

Figures 16a and 16b are "confusion matrices" or the distribution of errors for each of the algorithms. As can be seen, several types of errors are prevalent. For instance, with the continuous correction algorithm, 12 fours were identified correctly and 18 were misclassified as nines. Also, ten twos were identified as sevens. With the error correction algorithm, ten threes and 21 fives were thought to be eights. The "tie" column with the error correction algorithm indicates instances in which two output columns were exactly the same so that no classification could be made.

Again the error correction algorithm is superior to the continuous correction technique. It also required less computer time - six minutes for one iteration versus $10\frac{1}{2}$ minutes for the continuous correction algorithm. It is expected from the theory developed that the error correction algorithm would be superior in instances where the number of inputs in the organizing set is more than the number of dimensions of each input, 144 for these experiments.

Performing more than one iteration with 500 inputs is a difficult task in this computer simulation. There is not enough storage space in the computer to store the program and working data and also the data from all 500 input cards. Thus each card must be operated upon individually. Once the 500 cards have been used to adapt the matrix they cannot be reused for recognition. Thus two complete sets of cards were used. More iterations were possible with 100 inputs because these could be copied onto a work tape and the tape rewound many times. The data from 500 inputs is too much for one input tape and excessive time is used if several tapes were employed. The one case in which two iterations were

Figure 16. Confusion matrix for an experiment with 500 hand-printed inputs after one iteration. The table presents the distribution of errors for the experiment

Continuous correction algorithm

Input \ Recognized as	1	2	3	4	5	6	7	8	9	0	Tie
1	49						1				
2	11	12	7	1		2	10	3		4	
3	9	1	23		2		9	3		3	
4	9		1	12		1		7	18	2	
5	6				25	7		6	2	4	
6	7				1	34		3	4	1	
7	4						45		1		
8	11						2	25	4	8	
9	7						10		26	7	
0	2					2	6	1	2	37	

a)

Error correction algorithm

Input \ Recognized as	1	2	3	4	5	6	7	8	9	0	Tie
1	39						3	5			3
2		34	1	1	1	1	6	3	1	1	1
3	2	2	22			1	2	10	2	3	6
4				21		5	1	2	18		3
5		1	1		18	1		21		1	1
6		2			1	39		3	4		1
7	1		1				41	1	5		1
8		1	1			3		34	8	2	1
9	1					2	3	2	41		1
0		1	1	5	1	2	1	2	3	34	

b)

made with the 500 inputs was accomplished by outputting the 1440 weighting matrix values, w_{ij}, and then setting the matrix to these initial values for another iteration. About 20 minutes of computer time was required for this entire operation. These are the reasons more experiments were not performed on the complete set of 500 inputs.

Another technique which can be considered in the experiments is the introduction of a rejection criterion or threshold. In using a threshold, or discrimination level, a decision is rejected as unreliable if the difference between the largest and second largest column outputs is not greater than the discrimination level, d. Figure 17 is a plot of discrimination level versus rejection rate and error rate for each training algorithm with 500 inputs. Table 6 presents the data in tabular form.

Table 6. Error and rejection rates versus discrimination level for 500 hand-printed inputs after one iteration.

Algorithm used	Discrimination level, d	Error rate percent	Rejection rate percent
	.00	42.6	0.0
	.01	39.8	4.0
Continuous	.02	38.0	7.0
	.03	36.6	9.0
correction	.04	34.8	11.2
	.05	33.0	13.0
	.00	31.8	3.6
	.01	28.2	9.6
Error	.02	24.8	15.4
	.03	22.6	20.8
correction	.04	20.4	25.0
	.05	20.0	27.0

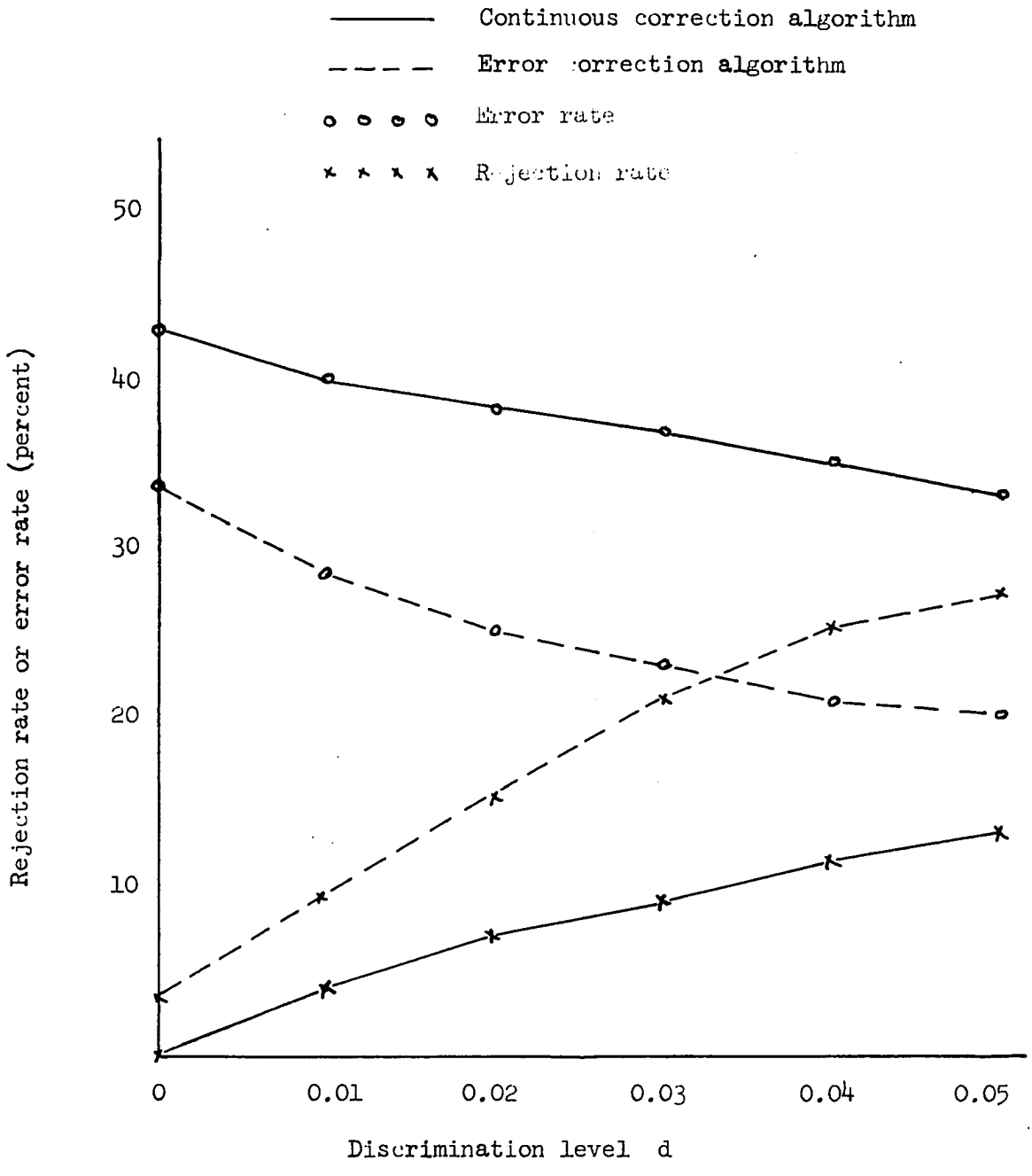


Figure 17. Error rate and rejection rate versus discrimination level for 500 hand-printed inputs after one iteration

Note that the error rate can be reduced but at the rather large expense of increasing the rejection rate. Hopefully the use of the rejection level would reject more decisions in which a mistake has been made than decisions which were correct. For the experiments demonstrated here, the use of a rejection zone does not appear to be justified.

Discussion

A comparison of the results reported here and results of other pattern recognition methods operating on the same data can be made. The actual recognition rates obtained in the computer simulation are not as important as the verification of the training algorithms proposed in the theoretical development. The actual performance of any categorizing method depends strongly upon the data upon which it operates. Complete comparison of the results reported here and other published results using this same input data is not possible since a recognition rate for a given experiment using the learning matrix has significance only when discussed in terms of the number of training iterations completed.

However, all other studies report recognition rates of the machine-printed data in the order of 99 percent, the same as reported here. Results with the hand-printed data vary from 83 percent in Highleyman (8) to 97 percent in Chow (1). Highleyman uses a "template matching" or correlation technique in which all the input data is examined and a weighting vector is found for each pattern class. The highest rate of 97 percent reported by Chow uses a method called "neighbor dependence". With this method, which is based on statistical decision theory, computations must be made to find the conditional probability that a point of the

12 x 12 matrix is covered if any of its neighbors is covered. The mechanization of such a network would be extremely difficult and the simulation required considerable computer time. Neither of these methods has the advantage of being adaptive. That is, they require all the input data to be present at the beginning; then the weighting factors are computed and are not changed thereafter. A big advantage of the iterative algorithms of the learning matrix is that each input is considered sequentially. Recognition can be attempted at any time in the training routine and additional training may easily be completed at any time. A relatively simple implementation of the learning matrix is also possible as described in the previous chapter.

A trade off must be employed in any pattern recognition system between sophistication and recognition rate. A simple system will not perform as well as a complex one. Although the 64.6 percent rate achieved by the learning matrix after one iteration on 500 inputs is lower than that of other methods, the rate would improve with further training. The experiments on 100 inputs indicate that this would be the case. The proof of the previous chapter (Theorem 10) that the error correction algorithm will find a solution if one exists is further evidence of improvement.

Summary

The results of the simulation experiments indicate the following conclusions can be made.

1) The learning matrix increases its ability to classify patterns as it receives additional training. Recognition rates on 100 inputs in-

creased from 77 percent to 95 percent in five iterations using the continuous correction algorithm and from 76 percent to 98 percent using the error correction algorithm.

2) The learning matrix is a particularly effective pattern categorizer if the input patterns are relatively uniform. For example machine-printed numbers were recognized with 99.2 percent accuracy even on numbers not members of the organizing set.

3) The learning matrix classified poorly (25-45 percent) hand-printed numbers not members of the organizing set. This is an indication that more inputs are needed so that the organizing set better represents the pattern class.

4) The error correction algorithm was superior to the continuous correction algorithm in every instance. It also took less computer time to simulate.

5) Use of a discrimination level with the learning matrix is not justified. The error rate can be reduced, but at the rather large expense of rejecting many inputs.

It is felt that the computer simulation fulfilled its purpose since the algorithms for training the learning matrix were found to operate sucessfully on a difficult, practical, pattern recognition problem.

SUMMARY AND CONCLUSIONS

This dissertation has considered a particular type of categorizer for pattern recognition applications called a learning matrix. This device is a matrix-like circuit structure in which the connections between input rows and output columns are variable parameters. The matrix is said to "learn" since its performance in a pattern recognition experiment improves over a sequence of trials.

A review and classification of pattern recognition methods has been presented. These are divided into three categories called correlation coefficients, linear decision functions, and statistical decision theory.

Two training algorithms for adapting the learning matrix have been developed. The first of these, called "continuous correction" adapts the learning matrix after each input and attempts to produce binary column outputs of one or zero. It is proved for this algorithm that the learning matrix will converge to a desired value if and only if the vectors of the input set are independent. Inputs are assigned to a pattern class by a "peak detection" method. That is, an input is assigned to the class associated with the maximum column output of the matrix.

The other training procedure is the "error correction" algorithm. With this method, the learning matrix is adapted only if a given input is misclassified in a peak detection sense. It is proved that this training procedure will converge in a finite number of adaptions to a matrix which will properly classify any number of inputs, provided a perfect classification is in fact possible.

A geometrical interpretation of the algorithms is given in terms of hyperplanes which separate each pair of pattern classes. If a discrimina-

tion or rejection criterion is used, the class-pair separation is by a pair of hyperplanes, symmetrically located about the origin.

A digital computer simulation of the learning matrix has furnished numerical results of the use of the learning matrix in an actual pattern recognition problem. The patterns which were recognized consisted of 50 sets of handwritten and machine-printed numbers which were originally prepared by Dr. W. H. Highleyman formerly of Bell Telephone Laboratories. The learning matrix properly classified 99.2 percent of the 500 machine-printed numbers after only one training iteration with these same 500 inputs. The recognition rate on 100 hand-printed numbers increased from 77 percent to 95 percent in five iterations using the continuous correction algorithm. The recognition rate on the complete set of 500 hand-printed numbers increased from 57.4 percent to 60.0 percent in two iterations using the continuous correction algorithm. With the error correction algorithm, 64.7 percent were classified correctly after only one iteration.

Several possible physical implementations for the learning matrix are also presented. These are all based on magnetic phenomenon with a thin-film device holding perhaps the greatest possibility for actual mechanization.

Concerning the learning matrix the following conclusions are established.

1) The learning matrix when trained with the continuous correction algorithm will converge to a desired value provided the inputs in the organizing set are independent. However the convergence is in the form of an infinite matrix series so that to reach the final value an infinite number of iterations would theoretically be required.

2) The learning matrix when trained with the error correction algorithm will converge to a value which will perfectly classify the input patterns provided such a classification is in fact possible. An upper bound is derived for the number of adaptions which are required to attain this result.

3) The continuous correction algorithm is in general inferior to the error correction algorithm. This is because it cannot be proved that the learning matrix when trained with the continuous correction algorithm will ever perfectly classify a dependent organizing set, even if perfect classification is possible.

4) The results of the computer simulation indicate that the learning matrix works well on patterns which are not extremely noisy or variable such as the machine-printed numbers. With sufficient training it also does well on noisy patterns such as the handwritten data. However a large organizing set must be employed before a satisfactory generalizing ability of the matrix, i.e., the ability to properly classify patterns not in the organizing set, can be obtained.

5) Suitable magnetic devices have already been suggested in the literature which make the physical implementation of the learning matrix feasible. Further research and improvement on these devices must be completed before the implementation becomes entirely practical.

The results of this dissertation provide information necessary for the design and construction of an actual pattern categorizer based upon the learning matrix model. Additional studies which could be made include:

1) Development of some practical measure of the rate of convergence of the iterative technique,

2) Derivation of an algorithm which will produce nonlinear surfaces to separate pattern classes instead of the hyperplanes presented here,

3) Study of possible advantages of using binary variables of minus one and plus one instead of the one and zero used in this research,

4) Study of the feasibility of a learning matrix with ternary and higher order input variables, and

5) Study of networks of learning matrices in which the outputs of one matrix become the inputs to another.

LITERATURE CITED

1. Chow, C. K. A recognition method using neighbor dependence. IEEE* Transactions on Electronic Computers EC-11: 683-690. October, 1962.

2. Crafts, H. S. Design of a magnetic variable-gain component for adaptive networks. Western Electronic Show and Convention (WESCON) Proceedings [Paper] 6.2: 19. 1963.

3. Finkbeiner, Daniel T., II. Introduction to matrices and linear transformations. W. H. Freeman and Company, San Francisco, California. 1960.

4. Greenberg, H. J. and Konheim, A. G. Linear and nonlinear methods in pattern classification. International Business Machines Journal of Research and Development 8: 299-307. July, 1964.

5. Harmon, Willis W. Principles of the statistical theory of communications. McGraw-Hill Book Company, Inc., New York, New York. 1963.

6. Hawkins, J. K. Learning by threshold elements in cascade. Aeronutronics Research Laboratory (Newport Beach, California) Report No. TN-20081. 1961.

7. Hawkins, J. K. Self-organizing systems - a review and commentary. IEEE Proceedings 49: 31-49. January, 1961.

8. Highleyman, W. H. An analog method for character recognition. IEEE Transactions on Electronic Computers EC-10: 502-512. September, 1961.

9. Highleyman, W. H. Linear decision functions, with application to pattern recognition. IEEE Proceedings 50: 1501-1515. June, 1962.

10. Ishida, Haruhisa. A generalized learning network using adaptive threshold elements. Unpublished Ph.D. thesis. Library, Iowa State University of Science and Technology, Ames, Iowa. 1964.

11. Mattson, R. L. The analysis and synthesis of adaptive systems which use networks of threshold elements. Stanford University Electronics Laboratory Technical Report No. 1553-1. 1962.

---

*Institute of Electrical and Electronics Engineers

12. Novikoff, A. B. J. On convergence proofs for perceptrons. Symposium on Mathematical Theory of Automata Proceedings: 615-621. 1962.

13. Pohm, A. V., Allen, G. R., and Nilsson, J. W. A magnetic film adaptive decision array. International Conference on Nonlinear Magnetics (INTERMAG) Proceedings [Paper] 11.5: 1-6. 1964.

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

15. Rosenblatt, Frank. Perceptron simulation experiments. IEEE Proceedings 48: 301-310. March, 1960.

16. Rosenblatt, Frank. Principles of neurodynamics. Spartan Books, Inc., Washington, D. C. 1961.

17. Sebestyen, G. S. Decision-making processes in pattern recognition. The Macmillan Company, New York, New York. 1962.

18. Steinbuch, K. and Piske, U. A. W. Learning matrices and their applications. IEEE Transactions on Electronic Computers EC-12: 846-862. December, 1963.

19. Varga, Richard S. Matrix iterative analysis. Prentice Hall, Englewood Cliffs, New Jersey. 1962.

20. Winder, R. O. Review of six papers in threshold logic. IEEE Transactions on Electronic Computers EC-11: 717-718. October, 1962.

21. Winder, R. O. Threshold logic in artificial intelligence. In Artificial Intelligence: [special publication] S-142. pp. 107-128. Institute of Electrical and Electronics Engineers, Inc., New York, New York. January, 1963.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his major professor, Dr. D. W. Gade, for helpful suggestions, to Robert Sharpe of the Cyclone Computer Laboratory for help in programming the computer simulation, and to Dr. Robert Meany of the Department of Mathematics for suggestions concerning some of the mathematical proofs.

Additional appreciation is due the National Science Foundation for fellowship support which enabled the author to devote full time in the completion of this research.

APPENDIX

This appendix presents the proof of Theorem 8 stated on page 52 .

Theorem 8. Given an input matrix X of m linearly independent n-dimensional vectors, where $m < n$, the infinite matrix series

$W_f = B + AB + AAB + AAAB + \ldots = \left[\sum_{t=0}^{\infty} A^t\right] B$, where A and B are

are as defined in Equations 8 and 9, converge to a matrix W_f such that $XW_f = I$. This is true even though the infinite series $\sum_{t=0}^{\infty} A^t$ taken alone, diverges.

Proof. The proof is carried out in detail for an input matrix X of $2(m = 2)$ n-dimensional linearly independent vectors. The proof for more inputs follows in an exactly analogous manner but becomes extremely long due to the large number of matrices involved. For example the matrix AAB for three inputs would require the sum of 448 matrices.

To simplify some of the equations involved, the following notation is used.

$$Q_i = \frac{X_i'X_i}{|X_i|^2} \qquad P_i = \frac{X_i'Z_i^*}{|X_i|^2}$$

The proof follows tediously but directly by substituting A and B for the two vector case into the expression for W_f. This gives,

$$W_f = B + AB + AAB + \ldots$$

$$= [P_1 + P_2 - Q_2P_1] + [P_1 + P_2 - Q_2P_1 - Q_1P_1 - Q_1P_2$$

$$+ Q_1Q_2P_1 - Q_2P_1 - Q_2P_2 + Q_2Q_2P_1 + Q_2Q_1P_1 + Q_2Q_1P_2$$

$$- Q_2Q_1Q_2P_1] + AAB + AAAB + \ldots$$

This expression can be simplified by observing that

$$Q_1P_1 = \frac{X_1'X_1X_1'Z_1^*}{|X_1|^2|X_1|^2} = \frac{(X_1 \cdot X_1)X_1'Z_1^*}{|X_1|^2|X_1|^2} = \frac{|X_1|^2 X_1'Z_1^*}{|X_1|^2|X_1|^2} = P_1$$

and that

$$Q_2Q_2 = \frac{X_2'X_2X_2'X_2}{|X_2|^2|X_2|^2} = \frac{(X_2 \cdot X_2)X_2'X_2}{|X_2|^2|X_2|^2} = \frac{|X_2|^2 X_2'X_2}{|X_2|^2|X_2|^2} = Q_2.$$

Then

$$W_f = [P_1 + P_2 - Q_2P_1] + [-Q_1P_2 + Q_1Q_2P_1 + Q_2Q_1P_2$$

$$- Q_2Q_1Q_2P_1] + AAB + AAAB + \ldots$$

A pattern for the terms of W_f can now be discerned as

$$B = P_1 + P_2 - Q_2P_1$$

$$AB = Q_1Q_2P_1 + Q_2Q_1P_2 - Q_2Q_1Q_2P_1 - Q_1P_2$$

$$AAB = Q_1Q_2Q_1Q_2P_1 + Q_2Q_1Q_2Q_1P_2 - Q_2Q_1Q_2Q_1Q_2P_1 - Q_1Q_2Q_1P_2.$$

Rewritting these expressions in terms of the input vectors gives,

$$B = \frac{X_1'Z_1^*}{|X_1|^2} + \frac{X_2'Z_2^*}{|X_2|^2} - \frac{(X_1X_2')\,X_2'Z_1^*}{|X_1|^2|X_2|^2}$$

$$AB = \frac{(X_1X_2')^2X_1'Z_1^*}{|X_1|^4|X_2|^2} + \frac{(X_1X_2')^2X_2'Z_2^*}{|X_1|^2|X_2|^4} - \frac{(X_1X_2')^3X_2'Z_1^*}{|X_1|^4|X_2|^4}$$

$$- \frac{(X_1X_2')X_1'Z_2^*}{|X_1|^2|X_2|^2}$$

$$AAB = \frac{(X_1X_2')^4X_1'Z_1^*}{|X_1|^6|X_2|^4} + \frac{(X_1X_2')^4X_2'Z_2^*}{|X_1|^4|X_2|^6} - \frac{(X_1X_2')^5X_2'Z_1^*}{|X_1|^6|X_2|^6}$$

$$- \frac{(X_1X_2')^3X_1'Z_2^*}{|X_1|^4|X_2|^4}$$

Summing all terms of W_f gives,

$$W_f = \left[\sum_{k=0}^{\infty} \frac{(X_1X_2')^{2k}}{|X_1|^{2(k+1)}|X_2|^{2k}}\right] X_1'Z_1^* + \left[\sum_{k=0}^{\infty} \frac{(X_1X_2')^{2k}}{|X_1|^{2k}|X_2|^{2(k+1)}}\right] X_2'Z_2^*$$

$$-\left[\sum_{k=1}^{\infty} \frac{(X_1X_2')^{2k-1}}{|X_1|^{2k}|X_2|^{2k}}\right] X_2'Z_1^* - \left[\sum_{k=1}^{\infty} \frac{(X_1X_2')^{2k-1}}{|X_1|^{2k}|X_2|^{2k}}\right] X_1'Z_2^*. \quad (39)$$

or

$$W_f = (a)X_1'Z_1^* + (b)X_2'Z_2^* - (c)X_2'Z_1^* - (d)X_1'Z_2^*,$$

where a, b, c, and d are the scalar infinite series given in brackets in Equation 39. Since a matrix series converges if and only if each element of the series converges (see Varga (19)) it must now be shown that these scalar series do in fact converge.

$$a = \sum_{k=0}^{\infty} \frac{(X_1 X_2')^{2k}}{|X_1|^{2k+2} |X_2|^{2k}}$$

$$= \frac{1}{|X_1|^2} \sum_{k=0}^{\infty} \left[\frac{X_1 X_2'}{X_1 \; X_2} \right]^{2k}$$

Since (a) is a power series it will converge if the general terms is less than 1. Thus the series will converge if

$$\frac{X_1 X_2'}{|X_1||X_2|} < 1.$$

Or

$$X_1 \cdot X_2 = X_1 X_2' < |X_1| \, |X_2|$$

From Schwartz's inequality

$$X_1 \cdot X_2 \leq |X_1| \, |X_2|.$$

But the equality holds only if $X_1 = K X_2$ where K is some constant. Since the vectors are assumed to be independent the equality cannot exist. Thus

the general term of (a) is less than one and the scalar series converges. The proof of convergence for (b), (c), and (d) is identical to that for (a).

W_f therefore converges. To prove $X W_f = I$, substitute Equation 39 into $X W_f$. Then

$$X W_f = X (a X_1'Z_1^* + b X_2'Z_2^* - c X_2'Z_1^* - d X_1'Z_2^*)$$

$$= X[(aX_1' - cX_2')Z_1^* + (bX_2' - dX_1')Z_2^*]$$

$$= \begin{vmatrix} X_1(aX_1' - cX_2') & X_1(bX_2' - dX_1') \\ X_2(aX_1' - cX_2') & X_2(bX_2' - dX_1') \end{vmatrix} .$$

It must now be shown that

$$X_1(aX_1' - cX_2') = 1 \tag{40}$$

$$X_1(bX_2' - dX_1') = 0 \tag{41}$$

$$X_2(aX_1' - cX_2') = 0 \tag{42}$$

$$X_2(bX_2' - dX_1') = 1 \quad . \tag{43}$$

Substituting a and c into Equation 40 gives

$$aX_1X_1' - cX_1X_2' = \left[\sum_{k=0}^{\infty} \frac{(X_1X_2')^{2k}}{|X_1|^{2k}|X_1|^2|X_2|^{2k}}\right]|X_1|^2 - \left[\frac{1}{X_1X_2'}\sum_{k=1}^{\infty} \frac{(X_1X_2')^{2k}}{|X_1|^{2k}|X_2|^{2k}}\right]$$

$$X_1X_2'$$

$$= 1 + \sum_{k=1}^{\infty} \left[\frac{X_1 X_2'}{|X_1||X_2|} \right]^{2k} - \sum_{k=1}^{\infty} \left[\frac{X_1 X_2'}{|X_1||X_2|} \right]^{2k}$$

$$= 1.$$

The proofs for Equations 41, 42, and 43 follow in a similar manner to that for Equation 40, thus proving Theorem 5.

As a demonstration of Theorem 8, consider an example. (Several have been worked out). Let

$$X = \begin{vmatrix} 1 & 1 & 0 \\ & & \\ 1 & 1 & 1 \end{vmatrix} \quad \text{Then } A = \frac{1}{6} \begin{vmatrix} 3 & -3 & -2 \\ -3 & 3 & -2 \\ 0 & 0 & 4 \end{vmatrix} \quad \text{and } B = \frac{1}{6} \begin{vmatrix} 1 & 2 \\ 1 & 2 \\ -2 & 2 \end{vmatrix}.$$

Then

$$AA = \left(\frac{1}{6}\right)^2 \begin{vmatrix} 18 & -18 & -8 \\ -18 & 18 & -8 \\ 0 & 0 & 16 \end{vmatrix}, \qquad AAA = \left(\frac{1}{6}\right)^3 \begin{vmatrix} 108 & -108 & -32 \\ -108 & 108 & -32 \\ 0 & 0 & 64 \end{vmatrix}$$

and

$$\sum_{k=0}^{\infty} A^k = \begin{vmatrix} \infty & -\infty & -1 \\ -\infty & \infty & -1 \\ 0 & 0 & 3 \end{vmatrix}$$

so the series diverges.

But

$$AB = \left(\frac{1}{6}\right)^2 \begin{vmatrix} 4 & -4 \\ 4 & -8 \\ -8 & 8 \end{vmatrix} \quad \text{and} \quad AAB = \left(\frac{1}{6}\right)^3 \begin{vmatrix} 16 & -16 \\ 16 & -16 \\ -32 & 32 \end{vmatrix}.$$

Then

$$B + AB + AAB + AAAB + \ldots \quad W_f = \begin{vmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ -1 & 1 \end{vmatrix}$$

so that the series postmultiplied by B converges. Then

$$X\,W_f = \begin{vmatrix} 1 & 1 & 0 \\ & & \\ 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & 0 \\ -1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ & \\ 0 & 1 \end{vmatrix} = I,$$

which completes the proof of Theorem 8.