SURVEY OF PATTERN RECOGNITION

Nils J. Nilsson

Artificial Intelligence Group
Stanford Research Institute
Menlo Park, Calif.

INTRODUCTION

One of the important uses of computers in clinical medicine is for the classification or screening of data. Examples abound where high speed and inexpensive but reliable automatic classification is desired. Electrocardiographs must be classified as healthy or abnormal. Differential white blood cell counts require the ability to discriminate between the various types of cells. Cancer-detecting smears must be sorted as normal or abnormal. It has become commonplace to speak of these kinds of sorting tasks as pattern-recognition problems and to advocate the application of pattern-recognition techniques for their solution. A wide range of such techniques exists. Some of them have been used by statisticians for years; others have been developed only recently as a result of the availability of high-speed computers. In this paper I shall describe some of the more common pattern-recognition methods.

Although I shall cite some clinical applications of pattern recognition as illustrative examples, it is not my purpose to report on these in detail. An extensive literature[1–3] already exists that provides numerous examples of the successful use of these methods. Indeed, many of the papers in this monograph will report the results of automatic classification experiments. I only hope that I can explain in this paper some of the unifying ideas that underlie many of the pattern-recognition methods already being applied. (See also a very good introductory article by Rosen.[4])

In order to apply pattern-recognition techniques, the phenomenon to be classified must be represented in some "computer-acceptable" form. Furthermore, the representation method used depends critically on the type of phenomenon. Thus, for photomicrographs of chromosomes, we might use complex picture-processing methods to represent the picture as a list of numbers, whereas, for a medical history record, it may only be possible to represent the data on the form as a list of nonnumerical symbols.

The phenomenon to be classified is called the event and its representation is called the data in order to distinguish between them. In this paper, I shall be primarily concerned with those pattern-recognition techniques for sorting numerical data, that is, representations that are in the form of a list of numbers. First, I shall describe several data-classifying methods (assuming that the event

has been appropriately represented as data) and then I shall retrace to discuss some examples of the way in which events can be represented as data.

Methods for Classifying Data

Fundamental Ideas

Suppose the event we wish to classify can be represented by a set of d numbers. Let the values of these numbers be denoted by the symbols $x_1, x_2, \ldots, x_d$. We shall call such a set of values a pattern. It is convenient to think of a pattern as a point $\vec{X}$ in d-dimensional space. The coordinates of the point are the values $x_1, x_2, \ldots, x_d$. Alternatively, it will sometimes be convenient to think of the pattern by the vector $\vec{X}$ with components $x_1, x_2, \ldots, x_d$.

Once an event has been represented by the pattern $\vec{X}$, we can speak of the problem of classifying the point $\vec{X}$ and hence the event. Suppose the event can belong to one of R categories. Now if all of the events belonging to a single category produced exactly the same set of d numbers, classification would be simple. A unique set of d numbers or vector $\vec{X}$ would correspond to each category. Classification of any event would then entail only a determination of which one of R unique vectors was identical to the vector representation of the event.

Unfortunately, our methods for numerical representation of events are not sophisticated enough to produce the same vector for every instance of an event of a given category. There is always a scatter of vectors in d-dimensional space representative of each class of event. In many pattern-recognition problems of practical interest, this scatter is quite extensive indeed, and the vectors representing different event categories may or may not intermingle.

As an example, consider the two-dimensional patterns illustrated in Figure 1, in which instances of white blood cells are represented by the values of two numerical parameters, cytoplasmic area and nuclear-cytoplasmic contrast. (The latter parameter is defined as the difference in optical density between the nuclear and cytoplasmic areas.) Note the scatter of points belonging to each of four types of cells: neutrophils, eosinophils, lympocytes, and monocytes. Also, note the scatter overlap between the monocytes and lymphocytes.

Most attempts to formalize the pattern-classification problem[5] begin by proposing techniques to deal with this scatter of pattern points. One fundamental assumption made about the representation method is that even though the pattern points are scattered, pattern points that are "close" together tend to belong to the same category. This assumption does not necessarily imply, however, that all of the patterns belonging to a given category are close together. (The distance by which "closeness" is measured is usually Euclidean distance.)

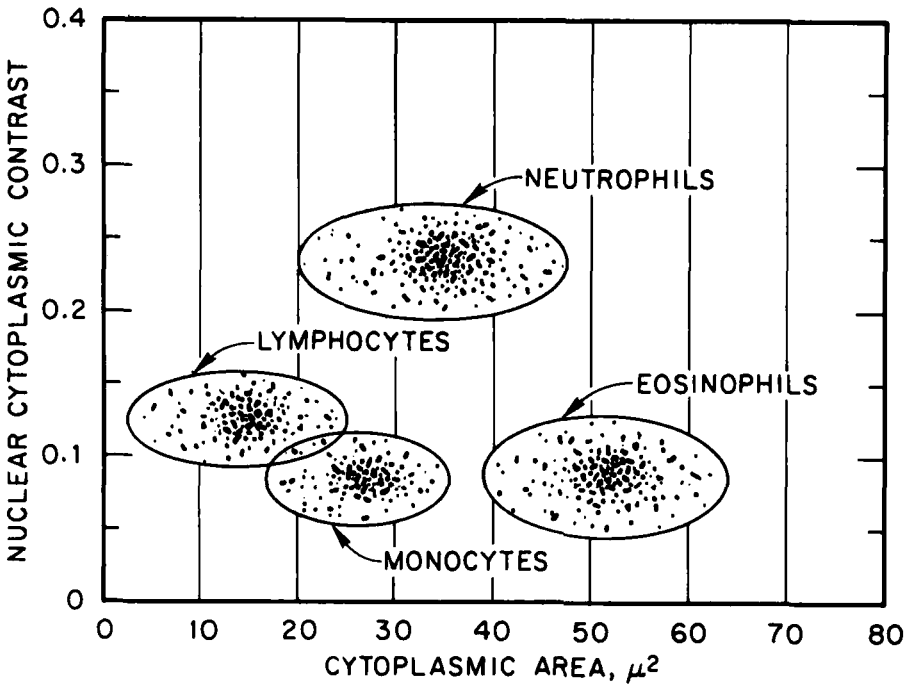

FIGURE 1. White blood cells represented by two parameters. (Based on FIGURE 9 in Prewitt & Mendelsohn).[37]

Given this assumption, it appears useful to employ the statistical notion of a probability-density function to describe the scatter of pattern points. Thus, let us assume that the patterns belonging to any category, say i, are random variables governed by a probability-density function $p(\vec{X}|i)$.

Once these probability-density functions are known, straightforward statistical analysis can be used to derive optimum methods for deciding to which category any given pattern $\vec{X}$ should be assigned. The optimum method is most often defined as that method which minimizes the probability of classifying a pattern in error. It is easy to show[5] that in this case the optimum method for classifying a pattern $\vec{X}$ calls for computing the quantities

$$p(\vec{X}|i)\ p(i) \text{ for } i = 1, \ldots, R$$

and assigning $\vec{X}$ to that category i_0 corresponding to the largest of these quantities; $p(i)$, $i = 1, \ldots, R$ are the a priori probabilities that $\vec{X}$ belongs to category i.

Consider, as an example, the three probability-density functions describing the scatter illustrated in FIGURE 2. There we have assumed the a priori probabilities $p(1) = p(2) = p(3) = \frac{1}{3}$ so the boundaries between classification regions are determined solely by the intersections of the three probability surfaces. The boundary lines represent the classification rule: Any pattern point to be classified is tested against these boundaries to determine whether it shall be called an instance of category 1, category 2, or category 3.

When the scatter of patterns belonging to a single category is governed by a multivariate normal distribution (an ideal situation, unfortunately not always seen in practical classification problems), the procedure for optimum classification can be thoroughly described by analytical methods.[6] I shall state the general procedure here and then mention some special cases.

Let us assume that for each $i = 1, \ldots, R$, $p(\vec{X}|i)$ is normal with covariance matrix K_i and mean vector $\vec{P}_i$. Then the optimum classifier (minimum probability of error) computes the functions:

$$g_i(\vec{X}) = -\tfrac{1}{2}\,\vec{X}^t K_i^{-1} \vec{X} \quad \text{(second-order terms)}$$

$$+\,\vec{X}^t K_i^{-1} \vec{P}_i \quad \text{(linear terms)}$$

$$-\tfrac{1}{2}\,\vec{P}_i^t K_i^{-1} \vec{P}_i + \log p(i) - \tfrac{1}{2} \log |K_i| \quad \text{(constant terms)}$$

for $i = 1, \ldots, R$, where

K_i^{-1} is the inverse of K_i,
$\vec{X}$ is a column vector,
$\vec{X}^t$ is the row vector form of $\vec{X}$, and
$|K_i|$ is the determinant of K_i.

The $g_i(\vec{X})$ in this case are quadric functions. They contain second-order terms such as $a_{ij}x_ix_j$, linear terms such as b_ix_i, and constant terms. The optimum classifier assigns $\vec{X}$ to that category i_o corresponding to the largest $g_i(\vec{X})$. This decision rule is equivalent to partitioning the pattern space with quadric (second-degree) surfaces. These surfaces separate those patterns $\vec{X}$ which are assigned to one category from patterns assigned to the other categories.

The example illustrated in FIGURE 2 is, in fact, one with three multivariate normal distributions. The second degree boundary surface in this two-dimensional case is a hyperbola.

Some special cases are worth considering. Suppose first that the scatter of points for each category has the same shape. For the normal distribution we

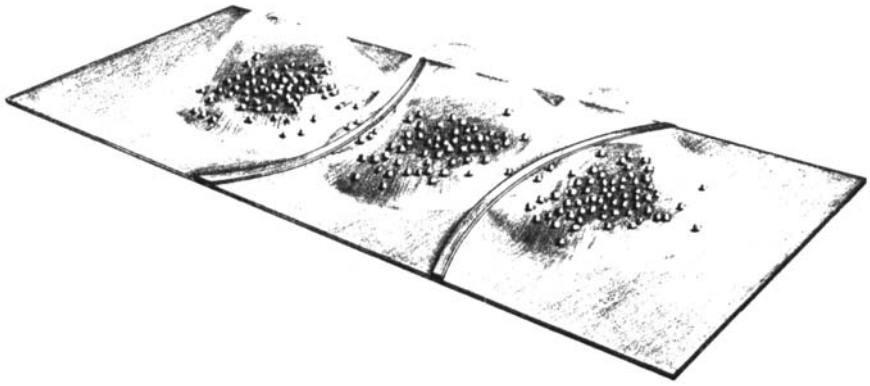

FIGURE 2. Three probability-density functions and the resulting classification boundaries.

mean then that each category has the same covariance matrix, K. In this case, the $g_i(\vec{X})$ are linear functions:

$$g_i(\vec{X}) = \vec{X}^t K^{-1} \vec{P}_i \quad \text{(linear terms)}$$

$$- \tfrac{1}{2} \vec{P}_i^t K^{-1} \vec{P}_i + \log p(i) \quad \text{(constant terms)}.$$

The boundary surfaces between any pair of categories are also linear. Such surfaces are called hyperplanes. Their orientation is determined by the location of the means and the shape of the scatter about each mean.

Another, very simple, special case results in a simple classification rule. Suppose the a priori probabilities are all equal and the scatters are all of the same magnitude and spherical (i.e., completely symmetrically distributed about the center). Then the $g_i(\vec{X})$ for the optimum classification method are:

$$g_i(\vec{X}) = \vec{X} \cdot \vec{P}_i - \tfrac{1}{2} \vec{P}_i \cdot \vec{P}_i$$

where $\vec{X} \cdot \vec{P}_i$ is the inner or dot product of $\vec{X}$ and $\vec{P}_i$. In this case, the pattern space is partitioned by hyperplanes that bisect the line segments joining pairs of mean vectors. This partitioning seems reasonable in this simple instance in which the scatter of patterns is spherically symmetrical and of the same extent for each category. A two-dimensional illustration for a four-category classification problem is shown in FIGURE 3.

Let me discuss some of the inadequacies of the multivariate normal model just presented and propose some possible remedies. In the first place, the exact probability densities can seldom be deduced a priori. We must usually observe a large sample of patterns resulting from many events in each category to get

an idea of the nature of the scatter. So even if we could know a priori that the form of the density functions was normal, we would still have to estimate from pattern samples the mean vector and covariance matrix for each category in any given pattern-recognition problem.*

Our methods for representing events numerically do not produce scatters of normal form in all cases of interest, however. For example, since there are

*See Abramson and Braverman[7] and Keehn[8] for discussions of methods for estimating mean vectors and covariance matrices.

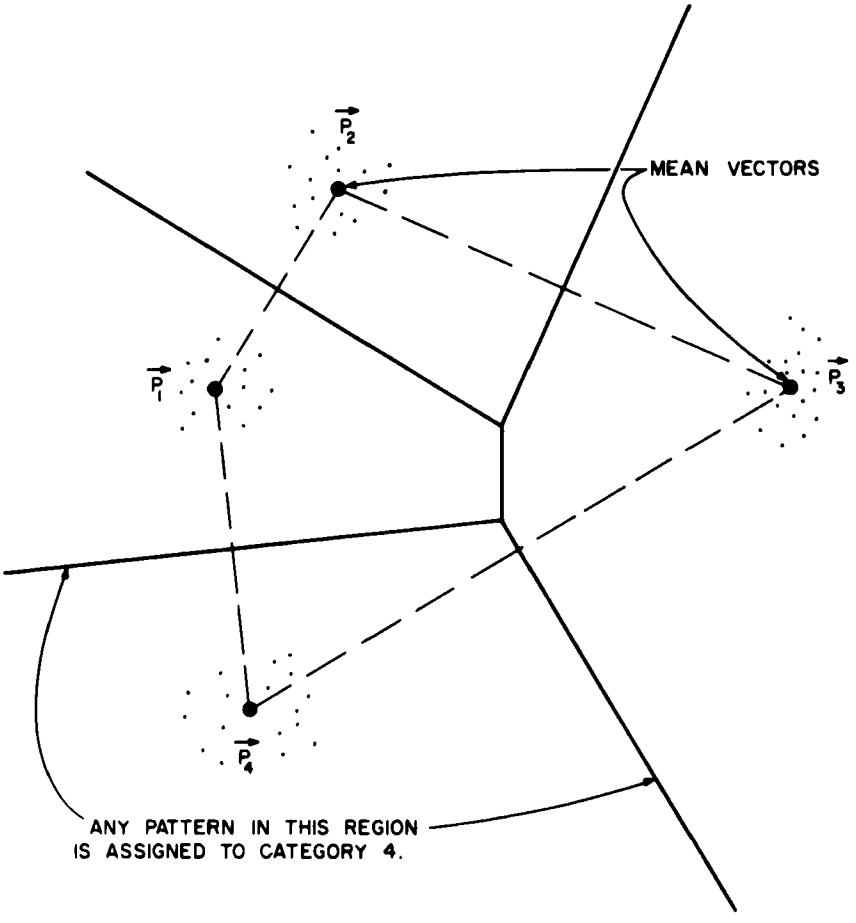

FIGURE 3. Four-category classification problem with spherically symmetrical normal scatters.

many different subclasses of cardiac malfunction,[9] the sample values of abnormal EKG waveforms are not normally distributed. Neither are the numerical representations of leukocytes based on integral geometric methods proposed by Brain.[10] In many problems, a more accurate density function would have many local maxima rather than the single one of the normal distribution. Nevertheless, even though many proposed representations do not produce normal scatters, the scatters, for all their complexity, are often separably distinct.

For these reasons, we need classification techniques that can simply estimate complex density functions from a large sample of patterns whose categories are known, and then use these densities in the computation of $p(\vec{X}|i)\ p(i)$ to assign patterns to the appropriate categories. The set of sample patterns on which the density estimates are based is often called the design set.

The Fix and Hodges Method

In this and the following sections, several related classification techniques are discussed. These techniques attempt either to estimate directly the values of probability-density functions from a design set of sample patterns, or to circumvent the explicit estimation of density functions by some roughly equivalent classification procedure. A representative list is presented.

One method for classifying patterns based on estimates of the values of probability-density functions is the Fix and Hodges method.[11] The Fix and Hodges method uses the patterns in the design set to classify a new pattern, $\vec{X}$, as follows: Select an integer k, and collect the k design patterns closest to $\vec{X}$ ("closeness" can be measured by Euclidean distance). Suppose that of these k closest patterns, n_1 patterns belong to category 1, n_2 to category 2, . . . , and n_R to category R, and $n_1 + n_2 + \ldots + n_R = k$. Then assign $\vec{X}$ to that category, i_o, corresponding to the largest of these numbers.

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

Selection of the integer k is important in the application of the Fix and Hodges procedure. If k is too small, the resulting decision rule might be too sensitive to the particular spatial locations of the design patterns. If k is too large, the rule will not be sensitive enough to the actual variations of the unknown probability distributions with $\vec{X}$. If the design set is large, it has been shown that the Fix and Hodges method leads to the same classifications as would be made if the (unknown) probability distributions were known and

used. In general, the value of k should increase without limit with increasing N, if N is the total number of patterns in the design set. The value of k/N, however, should decrease toward zero with increasing N.

The Nearest-Neighbor Method

Even when k is extremely small, however, the Fix and Hodges method still leads to classifications comparing favorably with those made by the optimum (minimum probability of error) classifier in the limit of very large design sets. This comparison is still quite favorable even when $k = 1$. When $k = 1$, the Fix and Hodges method places an unknown pattern, $\vec{X}$, in the same category as that of the closest pattern in the design set. Such a rule results in what is called the nearest-neighbor method. The nearest-neighbor method can be shown to be reasonably effective compared with the optimum classifier. Specifically, Cover and Hart[12] have shown that if PE* is the theoretical minimum probability of error of the optimum classifier using the true (but unknown) probability-density functions, and if PE is the probability of error resulting from the nearest-neighbor method, then PE is bounded by

$$PE^* \leq PE \leq PE^* \left(2 - \frac{R}{R-1} PE^* \right)$$

in the limit of an infinitely large design set. Certainly, in the limit, PE is never greater than twice PE*. In this sense it may be said that half the classification information in an infinite design set is contained in the nearest neighbor.

The Fix and Hodges and nearest-neighbor methods appear superficially to be reasonable answers to the problem of pattern classification. They suffer one important drawback, however. To classify any pattern, $\vec{X}$, the distance between $\vec{X}$ and each of the patterns in the design set must be computed. If these computations are to be performed rapidly, each of the design patterns must be stored in some rapid-access memory. Because the methods work best when the number of design patterns is large, the rapid-access storage requirements are often excessive. This disadvantage has motivated a search for other methods that preserve some of the features of the Fix and Hodges classifier without requiring the individual storage of every design pattern in a rapid-access memory.

Nearest-Prototype Methods

Several alternative methods can be discussed at this point. One might decide to use the design patterns differently to estimate the values of the probability-density functions. For example, Sebestyen[13,14] has proposed a process which

synthesizes an estimate of each probability-density function by adding together a number of component multivariate normal density functions. The locations and extents of the component functions are determined from the design patterns by an iterative process which does not require rapid-access storage of all of the design patterns.

Alternatively, one might abandon the idea of obtaining an explicit estimate of the values of the probability-density functions. Instead, one could attempt to use the design patterns in a way which would lead to the same category decisions for most unknown patterns as those which would be made by the Fix and Hodges or nearest-neighbor rules. Such a procedure might be nearly equivalent to one based on estimating probability-density functions, even though it does not explicitly estimate them. These equivalent procedures are considered in more detail in the remainder of this section.

The first of these equivalent procedures can best be explained as a "scaled-down" nearest-neighbor rule. I shall call it the nearest-prototype method. Rather than storing all of the design patterns, we might store only a few typical patterns called prototypes. Each prototype selected for storage might actually represent many design patterns which cluster around it in the pattern space. Classification of an unknown pattern $\vec{X}$ is then accomplished by assigning it to the category of the closest prototype. If there are a sufficient number of prototypes, it is reasonable to assume that the prototype closest to $\vec{X}$ will usually be of the same category as the design pattern closest to $\vec{X}$. Thus, the nearest-prototype method will usually assign patterns to the same category as does the nearest-neighbor method. The problem now is to discover a method for finding a reasonable number of prototypes to represent the design patterns.

One simple method is to provide only one prototype for each category and set that prototype equal to the center of gravity of all of the design patterns in the category. In some classification problems, in which the scatter of patterns belonging to each category is small, one prototype per category adequately represents all of the patterns belonging to that category, and a single-prototype-per-category method may produce excellent results. As soon as we encounter situations in which a wide variety of events might belong to the same category, and representation methods are unable to respond with relative uniformity to this variety, it becomes important to provide several prototypes per category.

At the other extreme, if we are assuming that the design set is large enough to represent accurately the inherent scatter in the problem, the design patterns could not be so completely intermixed as to require as many prototypes as design patterns. If such were the case, we certainly could not expect that any other new patterns would be likely to lie close to any of the prototypes, and the category of the nearest prototype would probably be irrelevant anyway. In this case, the design set is certainly too small a set on which to base the design of a classifier. Thus, in realistic situations, we should be able to find some number

of useful prototypes smaller than the number of design patterns and greater than or equal to the number of categories.

Several procedures have been suggested to find adequate prototypes or centers of clusters of patterns. Okajima *et al.*,[9] Firschein and Fischler,[15] Ball and Hall,[16] Bonner,[17] and Mattson and Damon[18] are among those that have proposed methods. Ball has reviewed several of these in a survey article.[19] The basis of many of these cluster-seeking techniques is a simple process. First, a number of trial points are proposed as possible cluster centers. Since the process iteratively adjusts these trial points, let us call them cluster-seekers. The process consists of two steps:

1. For each cluster-seeker $\vec{P}_i$, the set Y_i of design patterns closer to $\vec{P}_i$ than to any other cluster-seeker is determined. These sets Y_i are thus disjoint and exhaustive.
2. For each set Y_i determined in Step 1, the center of gravity $\vec{P}_i'$ of the set is computed. Each cluster-seeker is then changed from $\vec{P}_i$ to $\vec{P}_i'$ and the process returns to Step 1.

To use this process in a workable cluster-seeking method requires the addition of a way to allow new cluster-seekers to be born and others to coalesce. The reader is referred to the literature for more details of the various methods.

On the Structure of Nearest-Prototype Classification Methods

Let me digress for a moment to discuss the types of boundaries employed by the nearest-prototype classification methods and the kinds of structures that implement these boundaries. Suppose the prototypes are given by the points $\vec{P}_1, \vec{P}_2, \ldots, \vec{P}_i, \ldots, \vec{P}_M$ where there are M prototypes distributed over the R categories of patterns ($M \geq R$). To classify a new pattern $\vec{X}$ by the nearest-prototype method, we must calculate the distance to every prototype and find the smallest distance. Finding the smallest squared distance results in the same assignment, so we can instead calculate the quantities:

$$|\vec{X} - \vec{P}_i|^2 \text{ for } i = 1, \ldots, M.$$

These quantities can be written

$$\vec{X} \cdot \vec{X} - 2\vec{X} \cdot \vec{P}_i + \vec{P}_i \cdot \vec{P}_i.$$

Now, to find the smallest of these is equivalent to finding the largest of the following expressions:

$$\vec{X} \cdot \vec{P}_i - \tfrac{1}{2}\, \vec{P}_i \cdot \vec{P}_i \text{ for } i = 1, \ldots, M$$

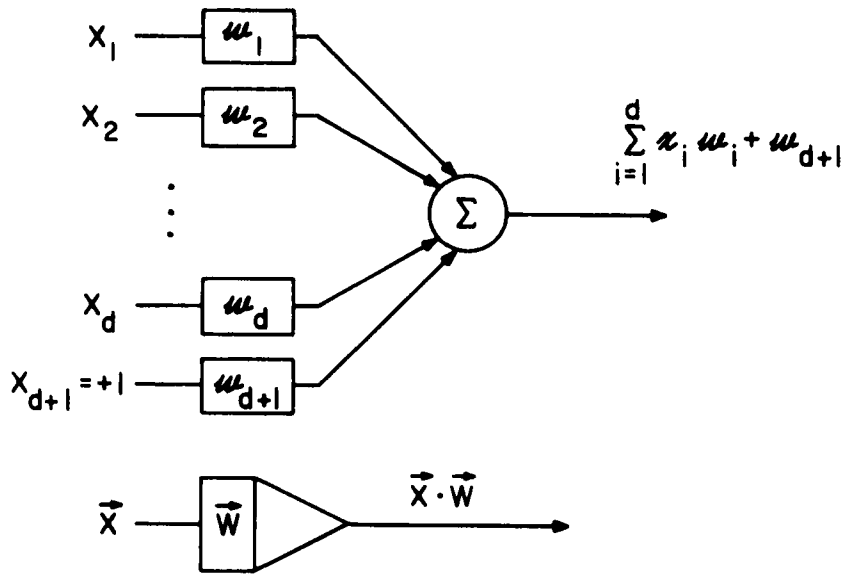

FIGURE 4. Dot product unit (DPU).

since $\vec{X} \cdot \vec{X}$ occurs in all of the expressions and multiplication by $-\frac{1}{2}$ reverses the ordering.

Therefore, we can employ very simple calculations in using the nearest-prototype method. The dot product of the pattern vector to be classified with each of the prototypes is computed, and a bias equal to one-half the squared length of each prototype is subtracted from each dot product.

Let us assume that each of these dot product calculations is computed by a device we shall call a dot product unit (DPU). The DPU is shown in FIGURE 4. A DPU may correspond to special-purpose electronic circuitry (e.g., a resistance adder) or to a digital computer subroutine. For patterns with d components, the DPU has (d + 1) "weights" $w_1, w_2, \ldots, w_d, w_{d+1}$ and computes a weighted sum,

$$S = \sum_{i=1}^{d} w_i x_i + w_{d+1},$$

of the pattern components. The first d weights correspond to the d pattern components, and the extra (d + 1)th weight allows a bias term to be added to the sum. It is often convenient to imagine that this (d + 1)th weight multiplies a fictitious (d + 1)th pattern component, which is set permanently equal

to some convenient value such as + 1. If the (d + 1) pattern components are represented by the (d + 1)-dimensional vector $\vec{X}$, and the (d + 1) weights are represented by the (d + 1)-dimensional vector $\vec{W}$, then the output of the DPU can be simply represented by the dot product $\vec{X} \cdot \vec{W}$. A classifier that employs DPU's is shown in FIGURE 5. In general, it may have M DPU's where $M \geq R$. For the nearest-prototype method, a DPU computation is made for every prototype. Each DPU has its first d weights set equal to the components of one of the prototype vectors and its (d + 1)th weight set equal to minus one-half the squared length of the prototype vector.

The surfaces in the pattern space which separate the patterns assigned to different categories by the nearest-prototype method are composed of segments of hyperplanes. These hyperplane segments are the perpendicular bisectors of line segments connecting prototype points belonging to different categories.

Error Correction Methods

Suppose we have a classification problem in which it is known that some techniques exist that can classify the patterns with a very low probability of error. For example, a trained clinician may be able to classify data representing

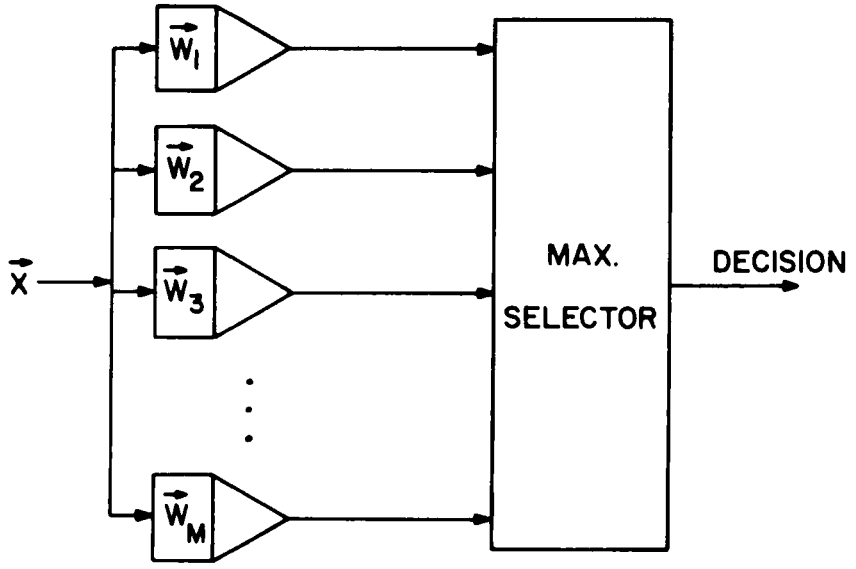

FIGURE 5. Classifier composed of DPU's.

white blood cells with 98 percent accuracy. Therefore, we know that the scatters representing each category cannot possibly overlap much, otherwise the clinician would have been incapable of such accurate classification. Whatever the probability-density functions might be, we know that the optimum boundaries separating the various classes are in regions of the space where patterns are sparse. Several methods exist which force "adjustable boundaries" into these sparse regions until the resulting classification rule makes an acceptably small number of errors on the patterns in the design set. Because the adjustments are prompted by errors made using trial boundaries, these methods are called error-correction methods.

Consider the classifier structure shown in FIGURE 5. It is called a piecewise-linear (PWL) machine because it implements boundaries which are composed of sections of hyperplanes. Since the PWL machine is such a simple structure, it is of interest to find methods for adjusting its boundaries until it classifies a representative sample of patterns in a design set with an acceptably small number of errors. The weight adjusting methods that I shall discuss involve the following steps: (1) selection of arbitrary initial weights of a PWL machine; (2) trial tests of the ability of this PWL machine to classify patterns in the design set correctly; and (3) adjustment of the weights in response to errors made by the PWL machine on the design patterns. For obvious reasons, methods employing these steps are called error-correction training methods.

Although an error-correction training method[5] has been proposed for a general PWL machine, I shall discuss a method for an important special case, the linear machine. A linear machine is a PWL machine with only one DPU for each pattern category (i.e., $M = R$). The linear machine is probably the most common of all pattern-classification devices. When its weights are set to implement a nearest-prototype classification method, there can be but one prototype per category. Even so it has been usefully employed in many recognition devices. When its weights are allowed to have arbitrary values, its range of usefulness is probably even greater. Highleyman[20] has discussed methods for setting the weights, as have Griffin, King, and Tunis.[21] Steinbuch and Piske[22] have employed essentially the same structure, calling it a learning matrix.

I shall discuss a particularly simple training method for the linear machine. First, the weights of the linear machine are set at arbitrary initial values. A pattern is then selected from the design set and tested on this initial version of the linear machine. If it is correctly classified, another pattern is selected and tested. When a pattern, say $\vec{X}$, is not correctly classified, the weights of two of the DPU's are adjusted. Suppose the $(d + 1)$-dimensional weight vectors prior to adjustment are denoted by $\vec{W}_1, \vec{W}_2, \ldots, \vec{W}_R$. Let the integer i be the actual category of the erroneously classified pattern, and let the integer $j \neq i$ be the category decision of the machine. Then only the weight vectors $\vec{W}_i$ and $\vec{W}_j$ are modified. Let their new values be given by the symbols $\vec{W}_i'$ and $\vec{W}_j'$.

The prescribed modifications in this case are

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

and

$$\vec{W}_j' = \vec{W}_j - c\vec{X}$$

where c is an arbitrary constant held fixed† at the same value during training.

This rule is a straightforward attempt to correct the error in the classification of $\vec{X}$. Obviously $\vec{W}_i \cdot \vec{X}$ was smaller than $\vec{W}_j \cdot \vec{X}$ when it was required that $\vec{W}_i \cdot \vec{X}$ be the largest output over all DPU's. Addition of $c\vec{X}$ to $\vec{W}_i$ will certainly increase the output of the ith DPU, and subtraction of $c\vec{X}$ from $\vec{W}_j$ will decrease the output of the jth DPU. These adjustments may or may not completely correct the error (depending on the value of c, relative to the magnitude of the difference between $\vec{W}_i \cdot \vec{X}$ and $\vec{W}_j \cdot \vec{X}$), but nevertheless they are steps in the right direction.

After a weight-vector adjustment, training continues by testing the linear machine on the design patterns, one at a time. The design patterns can be selected for test in fixed order, cycling through the set over and over, or in any random order that ensures that each pattern would be tested an infinite number of times if training were to continue for an infinite length of time.

Suppose it is known that there exists some setting of the weights of the linear machine that will correctly classify all of the design patterns. Then, the application of this error-correction training rule will, after only a finite number of corrections, produce a linear machine which does indeed classify all of the design patterns correctly. This result and its proof were first stated by Kessler (see Reference 5, Chapters 4 & 5). A subsequent proof by Duda and Fossum[23] covers a somewhat more general version of the rule. This rule has been applied successfully in many experiments. Of these I might mention the ones conducted by Duda and Fossum[23] and the one reported by Casey *et al.*[24]

A further interesting specialization of the PWL machine can be made. Consider the case of a linear machine for R = 2. Then there are only two DPU's. These compute the quantities

$$S_1 = \vec{W}_1 \cdot \vec{X}$$

and

$$S_2 = \vec{W}_2 \cdot \vec{X}.$$

†Actually it is possible to relax the requirement that c be fixed (see Reference 23). In some experiments better results are obtained if c is decreased slowly during training.

The linear machine in this case must decide only which is larger, S_1 or S_2. Such a simple comparison can also be made by testing to see if $(S_1 - S_2)$ is positive or negative. Let S denote the difference $(S_1 - S_2)$. S can be computed as follows

$$S = (\vec{W}_1 - \vec{W}_2) \cdot \vec{X}$$

or setting $\vec{W} = (\vec{W}_1 - \vec{W}_2)$

$$S = \vec{W} \cdot \vec{X}.$$

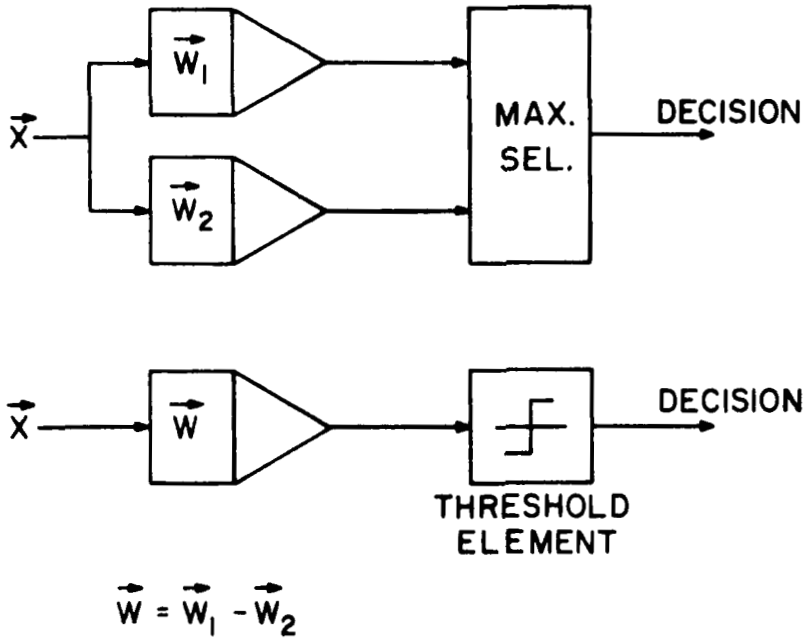

FIGURE 6. Threshold logic unit (TLU).

Therefore, for the R = 2 linear machine, a single DPU will suffice. This DPU can be followed by a "threshold element" to determine if $\vec{W} \cdot \vec{X}$ is positive or negative. Such a structure, called a "threshold logic unit" (TLU) is illustrated in FIGURE 6. The TLU is identical with the pattern-classifying device called ADALINE by Widrow.[25] It is also identical to the final decision element of the simple α-perceptron of Rosenblatt.[26] It divides the space of patterns into two classes by a single hyperplane. When the two-category classification prob-

lem is simple enough for such a surface to suffice, the TLU has been employed with excellent results.

The error-correction training rule for linear machines has a simple form for the TLU. If $\vec{X}$ is classified in error, one of two adjustments is made to the weight vector $\vec{W}$ at that stage. If $\vec{W} \cdot \vec{X}$ was erroneously negative (a category 1 pattern was assigned by the TLU to category 2), then the new weight vector is given by

$$\vec{W}' = \vec{W} + c\vec{X}.$$

If $\vec{W} \cdot \vec{X}$ was erroneously positive (a category 2 pattern was assigned by the TLU to category 1), then the new weight vector is given by

$$\vec{W}' = \vec{W} - c\vec{X}.$$

This rule and the proof of its convergence were first stated by Rosenblatt.[26] Its convergence was later also proved by Ridgway[27] and Novikoff.[28] Although we have presented it as a special case of the general rule for linear machines, the specific case (as usual) preceded the general by a few years. Actually, an adjustment technique very similar to the TLU rule was proposed by Agmon,[29] and Motzkin and Schoenberg[30] for the iterative solution of a set of linear inequalities several years earlier. The Agmon-Motzkin-Schoenberg rule turned out to be almost identical to a training process for the TLU proposed later by Widrow and Hoff[31] and called the minimum mean square error (MMSE) rule. Recently Koford and Groner[32] have shown that to the extent that the MMSE rule accomplishes its objective, the weight vector produced is identical to that prescribed for the optimum classifier when the patterns obey multivariate normal density functions with the same covariance matrix.

Polynomial Classification Surfaces

Error-correction classification techniques that have all used hyperplanes or surfaces composed of sections of hyperplanes as boundary surfaces to separate patterns belonging to different categories have been discussed. Techniques using more complex surfaces, such as surfaces described by polynomial equations, have also been employed in pattern-classification research. (Indeed, as we have already seen, second-degree or quadric surfaces are optimum for general multivariate normal probability-density functions.) It has been shown[5,33] that the use of polynomial surfaces can be considered theoretically as a simple extension of the use of linear surfaces. Error-correction training techniques can be used with "polynomial machines" with the result that in a finite number of steps the machine will make no errors on the training set if

the set is suitably separable by a polynomial surface within the power of the machine to implement. To use polynomial surfaces, however, one must calculate a large number of product terms such as $x_i x_j x_k$. To determine which technique is preferable in any given situation, the expense of making these computations as a step toward more complex surfaces must be compared with the expense of adding a few more DPU's to a PWL machine.

Representation of Events as Vectors

The Need for Ad Hoc Methods

There does not yet exist any general method for determining how various phenomena should best be represented as vectors for classification. So far each problem must be treated separately. Finding an appropriate numerical representation is largely an empirical matter following different *ad hoc* rules found to be useful in each special situation. For this reason, the design of a pattern-recognition system for classifying EKG signals, say, requires the active participation of a skilled cardiologist who is likely to know those aspects of an EKG waveform that are crucial to diagnosis. Similarly, the automatic classification of white blood cells requires the knowledge of a cytomorphologist. In fact, to put matters in their true perspective, it should be said that for almost any automatic recognition problem, 95 percent of the design effort involves the search for an appropriate numerical representation of the event to be classified. The recognition techniques that we have discussed so far in this paper, although interesting, should not be thought of as the most critical element of successful automatic classification procedure.

It would be impossible in a survey paper such as this to review all of the methods that have been used for representations. I shall discuss briefly here two simple examples and refer the reader to the literature for others. Notably, two earlier publications[34,35] of The New York Academy of Sciences on automatic classification, and the many papers on this subject in this monograph.

The examples come from two broad classes of classification problems. One, automatic classification of vectorcardiograms, illustrates phenomena whose primary data is in the form of a time-waveform. The second, automatic classification of white blood cells, illustrates phenomena whose primary data is in the form of a visual two-dimensional image.

Simple Examples of Vector Representation

The first example is based on a method for classifying vector-cardiograph waveforms as either normal or abnormal as reported by Specht.[36] The vectorcardiogram measured three components (right-left, head-foot, and anterior-

posterior), and the experiment was limited to an analysis of the "QRS complex" within the waveforms. Beginning after the onset of QRS, samples of each of the three waveforms were taken every five milliseconds up to 75 milliseconds, giving a total of 45 numbers. A 46th was added to represent the duration of the QRS.

Thus, each QRS complex was represented as a point in a 46-dimensional space. A classification method using polynomial surfaces was used to classify the 46-dimensional patterns. Even with such a simple representation method, classification accuracy was quite high. After training on 249 cases, the techniques correctly‡ classified 97 percent of 32 normals and 90 percent of 31 abnormals. Using information from the QRS complex alone, standard clinical diagnosis on the same cases achieved approximately 95 percent correct classification of the normals and only 53 percent correct classification of the abnormals.

The second example is based on an experiment conducted by Prewitt and Mendelsohn[37] on the classification of white blood cells from photomicrographs. Representation schemes for visual data is a large subject of its own. For representative techniques, the reader is referred to the monograph *Data Extraction and Processing of Optical Images in the Medical and Biological Sciences*.[35]

Prewitt and Mendelsohn's representation method involved essentially four stages:

1. Scanning, sampling, and digitizing to convert a 50 × 50 micron field of a photomicrograph into a 200 × 200 array of optical density measurements with 256 gray levels. This array is then stored in a digital computer.

2. Delineation of figure and ground to isolate a single white blood cell in the array.

3. Computation of a histogram of optical density values for the area in the array within the boundary of the cell.

4. Computation of as many as 35 parameters of the histogram. These parameters serve as the vector representation of the cell.

In one experiment, Prewitt and Mendelsohn showed that just three parameters derived from the histogram (nuclear area, cytoplasmic area, and nuclear-cytoplasmic contrast) were sufficient to permit classification of four types of leukocytes: neutrophils, eosinophils, lymphocytes, and monocytes.

Some Other Methods for Classification

The pattern-recognition methods which we have discussed in this paper might all be called parallel methods because each bases the classification of a

‡As determined by medical history records.

vector on the values of all of the components of that vector simultaneously or in parallel. An alternative method, which might be called a sequential or "tree" method, might look first at just one component of the vector, then look at some other component determined by the value of the first, and so on. Finally, the value of one of the components will determine a classification decision. Each test of a component narrows the possible classification decision until only one remains. Sequential methods are similar to the game of Twenty Questions, in which the answer to each question determines the next one to be asked. Parallel methods, in effect, ask all twenty questions at once and consider the answers in parallel to arrive at a decision.

Sequential methods are usefully employed when each of the component "questions" are answered consistently for all of the members of a given category. However, in cases where there can be a "scatter" of answers, sequential methods take too many "wrong turns." As a general rule, when the data are subject to scatter, it is usually wise to postpone any decision whatsoever until all the data are in, i.e., use parallel methods.

A limitation of the methods we have discussed is that they are all designed to work on numerical data only. Sometimes the event to be classified can best be expressed in some nonnumerical,§ symbolic form. The answers to a medical history questionnaire can generally be coded solely in symbolic form. Often it is convenient to represent a photograph in terms of a symbolic sequence of its parts. The parts themselves may be classifiable by our numerical methods, but the sequence of these classifications, being symbolic, demands other methods for its classification.

One approach to this problem is to list in a table all possible sequences of symbols together with the name or class we desire to assign to each one. Then when we want to classify a sequence, we look it up in the table to find its class. Obviously, this approach is often impossible because there would generally be an unmanageably large number of possible sequences. We are helped somewhat by the fact that typically many imaginable sequences are in fact impossible. (On a medical history form it is imaginable that a person might claim that his birthday was in 1940 and that he had the measles in 1935, but we would say that this sequence is illegal.) But even after ruling out all the illegal sequences, there are usually too many left. However, methods exist for economically representing large numbers of legal sequences. These are called syntax methods and involve specifying simple rules for constructing legal

§Of course, any symbol can be represented by a unique number and in this way symbolic data can be converted into numerical data. What we really mean by numerical data is that a "metric" like Euclidean distance exists in the space of patterns by which we can measure the "closeness" of patterns. All of the pattern-recognition methods we have treated depend on the rule: Patterns that are close together generally belong to the same category.

sequences out of symbol alphabets. Using these rules, straightforward methods exist for checking to see if an arbitrary sequence is legal, and if it is, what category of sequence it belongs to, but a full explanation of syntax methods is beyond the scope of this paper.¶

Conclusions

Several methods for classifying data that may be of use in clinical medicine have been discussed but pattern recognition should not be regarded as a magic new technique whose application will solve a lot of insoluble clinical problems. Rather, it consists primarily of a number of statistical methods for handling data that the digital computer now renders feasible. Furthermore, it is absolutely necessary that the clinician bring his specialized knowledge to the problem of applying any of these methods to a specific problem. In computer science there is a saying, "Garbage in, garbage out." Only the clinician can ensure that garbage is not going in. Then the computer scientist can struggle to see that none comes out.

References

1. Ledley, R. S. 1965. Use of Computers in Biology and Medicine. McGraw-Hill Book Co. New York, N. Y.
2. Stacy, R. W., & B. O. Waxman, Eds. 1965. Computers in Biomedical Research. Vols. I & II. Academic Press, Inc. New York, N. Y.
3. Whipple, H. E., Ed. 1964. Computers in Medicine and Biology. Ann. N. Y. Acad. Sci. 115 (2).
4. Rosen, C. A. 1967. Pattern classification by adaptive machines. Science 156 (3771): 38–44.
5. Nilsson, N. J. 1965. Learning Machines: Foundations of Trainable Pattern-Classifying Systems. McGraw-Hill Book Co. New York, N. Y.
6. Anderson, T. W. 1958. Introduction to Multivariate Statistical Analysis. John Wiley & Sons, Inc. New York, N. Y.
7. Abramson, N., & D. Braverman, 1962. Learning to recognize patterns in a random environment. IRE Trans. Info. Theory IT-8(5): 558–563.
8. Keehn, D. G. 1965. A note on learning for gaussian properties. IEEE Trans. Info. Theory IT-11(1): 126–132.
9. Okajima, M. *et al.* 1963. Computer pattern recognition techniques: some results with real electrocardiographic data. IEEE Trans. Bio-Med. Elec. BME-10(3).
10. Brain, A. E. 1967. The development of an automatic differential leukocyte counter. Internal Stanford Research Institute Memorandum. Menlo Park, Calif.
11. Fix, E., & J. L. Hodges, Jr. 1951. Discriminatory analysis, nonparametric discrimination: consistency properties. University of California Report 4 on Project 21-49-004, prepared under Contract AF 41(128)-37 for USAF School

¶An example of their application to chromosome classification can be found in Reference 1, pp. 339–344.

of Aviation Medicine, Randolph Field, Tex. (This report may be obtained through the Defense Document Center for Scientific and Technical Information, Cameron Station, Alexandria, Va.; refer to ATI 110633.)
12. COVER, T. M., & P. E. HART. 1967. Nearest-neighbor pattern classification. IEEE Trans. Info. Theory IT-13(1): 21–27.
13. SEBESTYEN, G. 1962. Pattern recognition by an adaptive process of sample set construction. IRE Trans. Info. Theory 178(5): 582–591.
14. SEBESTYEN, G. 1962. Decision-making Processes in Pattern Recognition. The Macmillan Co. New York, N. Y.
15. FIRSCHEIN, O., & M. FISCHLER. 1963. Automatic subclass determination for pattern-recognition applications. IEEE Trans. Elec. Comp. EC-12(2): 137–141.
16. BALL, G. H., & D. J. HALL. 1966. ISODATA, an iterative method of multivariate data analysis and pattern classification. Proc. Int. Comm. Conf. Philadelphia, Pa.
17. BONNER, R. E. 1966. Cluster analysis. Ann. N. Y. Acad. Sci. 128 (3): 972–983.
18. MATTSON, R. L., & J. E. DAMMAN. 1965. A technique for determining and coding subclasses in pattern recognition problems. IBM J. Res. Develop.: 294–302.
19. BALL, G. H. 1965. Data analysis in the social sciences: what about the details? Proc. Fall Joint Comp. Conf. 27 (I): 533–559. Spartan Books. Washington, D. C.
20. HIGHLEYMAN, W. H. 1962. Linear decision functions, with application to pattern recognition. Proc. IRE 50 (6): 1501–1514.
21. GRIFFIN, J., J. KING & C. TUNIS. 1964. A pattern identification device using linear decision functions. *In* Computer and Information Sciences. J. T. Tou & R. H. Wilcox, Eds. : 169–193. Spartan Press. Washington, D.C.
22. STEINBUCH, K., & V. A. W. PISKE. 1963. Learning matrices and their applications. IEEE Trans. Elec. Comp. EC-12(5): 846–862.
23. DUDA, R. O., & H. FOSSUM. 1966. Pattern classification by iteratively determined linear and piecewise linear discriminant functions. IEEE Trans. Elec. Comp. EC-15(2): 220–232.
24. CASEY, R. G., Ed. 1965. An experimental comparison of several design algorithms used in pattern recognition. IBM Res. Rep. No. RC1500.
25. WIDROW, B. 1962. Generalization and information storage in networks of ADALINE 'neurons.' *In* Self-Organizing Systems. M. C. Yovits, G. T. Jacobi & G. D. Goldstein, Eds.: 435–461. Spartan Books. Washington, D. C.
26. ROSENBLATT, F. 1961. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books. Washington, D. C.
27. RIDGEWAY, W. C. 1962. An adaptive logic system with generalizing properties. Stanford Electronics Labs Technical Report 1556-1, prepared under Air Force Contract AF 33(616)-7726. Stanford, Calif.
28. NOVIKOFF, A. B. J. 1963. On convergence proofs for perceptrons. Symp. Math. Theory of Automata, Brooklyn, N. Y.: 615–622. Polytechnic Press.
29. AGMON, S. 1954. The relaxation method for linear inequalities. Canad. J. Math. 6 (3): 382–392.
30. MOTZKIN, T. S., & I. J. SCHOENBERG. 1954. The relaxation method for linear inequalities. Canad. J. Math. 6 (3): 393–404.
31. WIDROW, B., & M. E. HOFF. 1960. Adaptive switching circuits. Stanford Electronics Labs Technical Report No. 1553-1. Stanford, Calif.
32. KOFORD, J. E., & G. F. GRONER. 1966. The use of an adaptive threshold element to design a linear optimal pattern classifier. IEEE Trans. Info. Theory IT-12 (1): 42–50.
33. COVER, T. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Trans. Elec. Comp. EC-14(3): 326–334.
34. WEYER, E. M., Ed. 1966. Advances in Bio-Medical Computer Applications. Ann. N. Y. Acad. Sci. 128 (3).

35. KRAUSS, M., Ed. Data Extraction and Processing of Optical Images in the Medical and Biological Sciences. 1969. Ann. N. Y. Acad. Sci. 157 (1).
36. SPECHT, D. F. 1967. Vectorcardiographic diagnosis using the polynomial discriminant method of pattern recognition. IEEE Trans. Bio-Med. Eng. BME-14(2): 90–95.
37. PREWITT, J. M. S., & M. L. MENDELSOHN. 1966. The analysis of cell images. Ann. N. Y. Acad. Sci. 128 (3): 1035–1053.