1969

Fixed classifier pattern recognition using iteratively produced preprocessing

Harold William Workman
Iowa State University

69-20,687

WORKMAN, Harold William, 1943-

FIXED CLASSIFIER PATTERN RECOGNITION USING ITERATIVELY PRODUCED PREPROCESSING.

Iowa State University, Ph.D., 1969
Engineering, electrical

University Microfilms, Inc., Ann Arbor, Michigan

FIXED CLASSIFIER PATTERN RECOGNITION USING ITERATIVELY PRODUCED PREPROCESSING

by

Harold William Workman

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
Ames, Iowa

1969

TABLE OF CONTENTS

I. THE PATTERN RECOGNITION CONCEPT

The pattern recognition problem is a study into the process of learning. It is also involved with associated processes. Of utmost importance is the process of hypothesis formation, hypothesis testing, and hypothesis correction. Another significant process is that of information gathering. Information gathering is important in that learning cannot take place if adequate information is not available. Learning can be hindered if unnecessary information is included so that it clouds and confuses the learner. How to use available memory is yet another problem which must be dealt with.

The pattern recognition problem is one of classifying patterns, or objects. The pattern recognizer is first given a set of patterns and also the classification of each pattern. Using the set as a guide, the pattern recognizer is to learn the characteristics of each class. After the learning is completed, the pattern recognizer must be able to classify correctly patterns which have not been encountered explicitly in the learning set. It is assumed beforehand that each pattern possesses only one classification, and that the identity of this classification is obtainable from some external source for at least a limited number of patterns so that the pattern recognizer may learn.

As an example, suppose that the series of patterns are a sequence of pictures. Suppose further that some of these pictures are photographs, some are paintings, some are block prints, and some are lithographs. Let each picture be one of these four classes, and only one. The pattern recognizer uses a set of pictures, for which the proper classification is known, to learn how to distinguish between paintings, photographs, block prints, and lithographs. After the learning process is completed, new and different pictures may be classified according to what was learned previously.

The set of pictures that was used to learn about the classifications is called the training or learning set. The set of pictures that is used after the learning process is completed is known as the functional or test set. The training set is identifiable as the training set because the correct classification of each pattern is presented at the same time the pattern is presented for learning. The test set is called the test set because this is the set that is used to determine how well the pattern recognizer has learned. It is called the functional set also because this is the set that is used when the pattern recognizer is performing a useful function, that of classifying patterns.

It is said that the pattern recognizer has learned if it performs better on the test set after examining the learning set. "Better" usually means that the percentage of correct

classifications has increased.

A. Features

First the object must be perceived and studied. Requiring the pattern recognizer to examine an object is an assumed operation. What the pattern recognizer examines about the object is not assumed, and is a crucial part of the problem. It must be assumed that the means are available to examine the object sufficiently well to make a correct decision.

To examine an object means that measurments are taken on it to determine what characteristics it possesses. Each of these measurements may be called a feature of the object. There are two fundamental types (16). The first type is the kind of feature that all members of the same class possess. For example, suppose that one is separating apples from grass. One might note that all apples have stems and all apples have a white pulp. These are known as intraset features. The second type of feature is that possessed by some members of the class, but not by some other class. For example, some apples are red, while grass is never red. Some apples are green like grass, however. These are called interset features. Notice that a feature may be both an interset and an intraset feature, or it may be one or the other, or neither.

Features that are intraset or interset are significant in

determining the class of an object.

A feature is a description, an adjective, or a combination of descriptions. A feature may be simple, such as "a black dot." It may be complex, such as a "crescent shaped, glowing object."

Every object of a class has at least one feature. That one feature is that it belongs to that particular class. This feature is both an intraset and an interset feature.

It should be pointed out that it is assumed that an object either has a certain feature or it does not have that feature. There is no in between. For example, a feature may be "blue or green." Once the feature is unambiguously defined with respect to all possibilities, an object has the feature or lacks it.

The pattern recognizer is interested in one and only one measurement. That measurement is the one feature that identifies that object as a member of its class. This feature is not known beforehand. It is very important to realize that the entire basis for thinking that a pattern recognition scheme will be successful is that this feature can be revealed by examining other features which are sufficiently correlated with the basic classification feature. If the features selected show no correlation with the classification feature, then a complete and perfect solution is not obtainable. The notion of a solution excludes the remote

possibility that a machine might correctly classify all the vectors in the test set by classifying objects in a random or chance fashion.

B. Hypothesis Formation

A hypothesis is a theory about why patterns belong to their classes and not to a different class. Suppose that a pattern recognizer merely memorized each and every member of the learning set and its associated class. The pattern recognizer would then do an excellent job of classifying if it were tested by giving it only patterns that belonged to the learning set. That is, if the test set was identical to the learning set. In most cases this is not the situation. Usually the test set will contain examples not contained identically in the learning set. If the objects were pictures, one would not expect all the possible pictures to be in the learning set. Only a few representative ones are present.

The only way that the pattern recognizer will be able to do well on the test set is to form some hypothesis or generality about the test set from the few examples present in the learning set. In some manner the pattern recognizer must extrapolate from the specific features or measurements which vary from pattern to pattern to a feature which is invariant for members of the same class. This is the crucial problem in the pattern recognition process.

II. THE PATTERN RECOGNITION SYSTEM

A pattern recognizer may be considered as part of a system. The possible success of the pattern recognizer is dependent upon the components of the rest of the system. Figures 1 and 2 show two possible systems. Figure 1 has an input signal which is the name of the class. The object generator generates a member of the particular class given at its input. The object generator may be influenced by noise. This noise signal influences the output of the object generator. An example of an object generator is a person who is told to make a particular letter of the alphabet. The output of this person may be a letter marked on a piece of paper with a pencil. The letter will vary from what the person desires to make, that is, the ideal representative of the class, by corruptive influences or noise. Possible causes which create variances in the letter are such things as the sharpness of the pencil, the roughness of the paper, the position of the letter on the paper, and what time of the day it happens to be.

As another example, consider a pattern recognizer which must detect shoplifters. A shoplifter, having decided beforehand to shoplift, generates a pattern. He may hide behind the store counters, have unusual packages, or have unwrapped packages. The pattern recognizer must detect these patterns

and conclude that the person is stealing.

Figure 2 shows another system. Here the object vector, or pattern, is the initial input to the system. An ideal pattern recognizer evaluates this vector and classifies it. The problem is to construct a pattern recognizer which is functionally identical to the ideal pattern recognizer.

As an example, consider a pattern recognizer that must learn to break a code. The learning set might be pairs of documents, each pair consisting of the true message and the same message in code. The code has a solution. That is, it is definitely known that somewhere an ideal pattern recognizer exists that can properly decode the message.

The differences between these two systems need to be examined. In Figure 1, a pattern recognizer needs to be constructed that is the functional inverse of the pattern generator. The solution to the problem depends greatly upon whether this inverse exits. That is, if the noise signal is severe enough so that two different inputs to the pattern generator produce the same pattern or patterns that differ only in their noise content, then a solution doesn't exist to finding the inverse of the generator.

Figure 2 has a guaranteed solution if the ideal pattern recognizer is deterministic. For a given input, it must always have the same output.

An interesting problem is the prediction problem. This

is the system of Figure 2, but with the stipulation that the pattern recognizer to be constructed must be better than the ideal pattern recognizer. Consider a fire detection and prevention system. The ideal classifier is a device which makes several measurements of its environment over a period of time, and sounds an alarm when fire erupts. The pattern recognizer to be constructed must reach the same conclusion concerning the possibility of a fire, but by using only the measurements made long before the fire will actually break out. That is, the pattern recognizer must predict the fire.

A block diagram of a pattern recognizer is shown in Figure 3. The input to the pattern recognizer is the object vector. The output is the best estimation as to what class of objects this input vector belongs. The output is a scalar quantity. The input vector is composed of elements of which each can take on only one of a finite number of states. That is, each element value is of a set with a finite cardinal number of members.

The system may be broken into four distinct portions. The one-to-one transformation, the preprocessing transformation, the feature extraction transformation, and the classification transformation. The first transformation which may be present is the transformation which does not reduce or change the information content of the input vector. The characteristic of these transformations is that they are

uniquely reversible. That is, if the output of the transformation is given, then the input vector can be uniquely identified. For any given input there can exist one and only one output, and for any given output there can be one and only one input which caused that output to occur.

An example of this type of transformation would be changing from a base 10 system to a base 2 system. That is, each number written in the base 10 system would be converted to a number in the base 2 system. This transformation is reversible because given the output, a binary number, only one number in base 10 system could have caused that base 2 number to have occurred.

The preprocessing and the feature extraction transformations both reduce the information content of the input vectors. The preprocessing transformation doesn't reduce the dimensionality of the input vector, while the feature extractor may. The preprocessor examines it's input for information which is known to be trivial in the decision process. It deletes this information from the vector so that this information will not lead the classifier to a wrong conclusion.

An example of this is "fly-speck" removal in character recognition. Suppose that the input was a letter or number that was handwritten. Sometimes blotches of ink may appear on the paper. Irregularities in the paper may also cause

extraneous dots to appear. The preprocessor senses that these ink spots are not necessary for the decision process, and may actually be a hinderance. Hence they are removed.

The feature extractor reduces the dimensionality of the input vector. The preprocessor does not. Only a small amount of significant information may be required for the classifier to classify the vector properly. The feature extractor extracts the important information from the input vector so that the effort required of the classifier is minimal. A great deal of information may be ignored and deleted by the feature extractor. For this reason, it is often the deciding factor for the success of the system. Suppose that the input vectors were faces of men and women (actually measurements made on them). If, among other extractions, the feature extractor extracted the presence of the whiskers on the face (by using combinations of other measurements), the classification of these faces as men or women would be greatly simplified.

The classifier takes the output of the feature extractor and makes a judgement as to which class it belongs. The classifier must ultimately judge the probability of being right for each class choice, and select the most likely. This is in accordance with classical decision theory. The classifier can also be thought of as a distance measuring instrument. It must measure the distance that each vector is

from each class possibility, and choose the closest class as the correct class. A distance of zero can occur if the classifier has encountered the input vector previously. The fact that the classifier must measure the distance in one particular way leads to a limitation in its performance. The transformations that are made to the object vector before it is given to the classifier are to help alleviate the classifier' limitations.

III. REVIEW OF THE LITERATURE

A. The Character Recognizer

Character recognition is a specific problem to which pattern recognition techniques are often applied. The problem is to recognize letters of the alphabet and numbers. The usual implication is that a reading machine is necessary to complete the man-machine interface. The reading of prespecified letters and numbers is not difficult, as can be evidenced by machine readable numbers on bank checks. However, the problem of reading handwritten letters and numbers is much more difficult, because no two people write exactly the same, and a large number of people create large variances from the normal sample.

Several schemes for recognizing numbers and letters are in the literature. Only a few will be discussed here. These schemes have the common feature that they were especially devised for recognizing letters and numbers. They do not contain elaborate learning programs, but rather the learning was primarily done by the designer as he prepared the reading machine. These schemes can be considered as the feature extractors and classifiers of Figure 3, after all learning has ceased. That is, the feature extractor-classifier is a fixed piece of hardware, and any given input will always produce the identical classification, independent

of time. It is the inability to learn further that tends to make these schemes be useful for only a limited type of input, but their cleverness and compactness make them desirable to try to invent.

A method proposed by Turner (17) is illustrative of a process called template matching. Turner describes a device to read printed characters as they move along in front of a sensing device. The sensing device is composed of photocells arranged so that as the letter moves across the field of view of the cells, the dark portions of the letter block the incident light to some of the cells. For each character to be recognized a weighting function is conceived so that the output of this function (a time varying voltage), is a maximum only if the letter that this function is associated with is directly under the sensing device. A function must be used for each class, and the system may be altered only by adding more functions. The weighting function can be thought of as a template, for which letters of the same class match.

Unger (18) describes a method which utilizes a feature extractor for the primary portion of the investigative process. The features extracted are utilized in the decision process by the use of a decision tree. A decision tree is equivalent to writing a Boolean function for each character, where the features extracted are the variables. The functions are

written so that only one character is identified for any given set of values in the functions.

Each character is represented by a grid of one's and zero's. The element of the grid is assigned a 1 if the letter written covers that element with pencil lead or ink. Otherwise it is assigned the value zero. Using a digital computer, Unger first preprocesses by making all lines of equal width. Calculations are made to determine if the edges of each letter are vertical or horizontal and in what order they occur. Also the relative lengths of each edge is noted. These measurements constitute the output of the feature extractor. They are then combined using the decision tree to make the final decision.

Sherman (13) also uses a decision tree as the classifier. His set of features that are extracted differ from Unger's. The basic extraction is the topographical nature of the figure. He notes the number of lines in the figure and the number and kind of intersection of these lines. These features are not sufficient, however, and additional information in the form of the angle of the lines is necessary. One of his problems is that different people may write the same letter in topographically different ways. As an example, the number 8 might be drawn in a manner that is topographically different, such as 8.

Sprick and Ganzorn (14) feature extract by following

the right and left boundaries of the number. A time varying voltage is generated which is equivalent to the distance the right side of the character is from a vertical reference line as the time base moves from the top of the character to the bottom along the right side. A similar measurement is made of the left side. These two measurements are deemed sufficient to recognize any number. Good success was found on printed alphabets.

Another technique that utilizes the natural curves of the figure is put forth by Greanias, et al. (5). They developed a sophisticated scanner that produced the equivalent of a time traced line that followed the outline of the number. After this basic feature was extracted, additional features were taken from it to make a decision tree. These features are similar to Unger's. The results can be considered good in that 92% of handwritten numbers could be correctly identified for untrained subjects, and 99.3% for trained writers after 30 minutes of training.

A number of common problems seem to recur continually for investigators. Usually, whenever possible, these are removed by preprocessing. Having a "clean" figure is often necessary. This means that stray marks on the paper need to be removed in some manner. The figure should have sharply defined boundaries and line values if accurate line derived features are to be obtained. Inaccuracies might result if

the figure is rotated slightly or translated.

These problems have lead some investigators to create pattern recognizers that are unaffected by translation, or perhaps by rotation, or perhaps by the size or by small smudges of ink (10). These approaches are not general, for the difference between two classes could be only a translation difference. For example, the difference between a "w" and an "m" is not easily found if the character recognizer is rotation invariant.

Tenery (15) suggests a method which is invariant to the position of the character or the rotation of the character. A probability function $P_F(d,r)$ is defined. $P_F(d,r)$ is the conditional probability that if some origin point A of a line segment of length d falls within a Figure F lying entirely within a boundary (viewing area), the terminal point B also falls within the figure. The point B is located at a distance d, a normalized constant, and at all angles of rotation r. Examination of the magnitude of the probability was used to discriminate between a limited pattern set.

Horwitz and Shelton (9) utilize an autocorrelation function to achieve translation invariancy. Their basic feature extraction is a function defined to be

$$D(R) = \sum_{r} f(r)f(r-R) \qquad 1)$$

where $f(r) = 1$ only if the coordinate value r describes a "black" square, and

$f(r) = 0$ otherwise.

$D(R)$ is the total number of pairs of "black" squares separated by r-R.

To determine the class identity for a vector, Horwitz compares the value of the function of the unknown vector A to the value of the function for each vector B of known identity by using a similarity function, S_{AB}.

$$S_{AB} = \frac{\sum_{R} D_A(R) D_B(R)}{(\sum_{R} D_A^2(R))^{1/2} (\sum_{R} D_B^2(R))^{1/2}} \qquad 2)$$

B. Pattern Recognition

Pattern recognition is basically a decision problem. It is therefore a candidate for the application of classical decision theory. Bayes' decision rule is usually taken as the best rule for decision making. Bayes' rule will be given here in simplified form.

1. Bayes' decision rule

Bayes' decision rule (8) is an attempt to minimize the risk involved with the decision. Suppose that the penalty for making a mistake is the same for all types of errors and that the penalty for making a correct classification is

zero.

Then the risk involved for making a particular decision is just the probability that the given input vector is actually a different class than the class chosen.

If the classes are A, B, and C, and if the input vector is 0, then the risk in choosing class A is

$$r(A) = p(B/0)+p(C/0) \quad 3)$$

and the risk involved in choosing class B is

$$r(B) = p(A/0)+p(C/0) \quad 4)$$

and likewise

$$r(C) = p(A/0)+p(B/0) \ . \quad 5)$$

Where the following general notation will be used.

A,B,C are class names.

p(A) is the probability of A occurring.

p(A/0,X) is the conditional probability of class A occurring, given that 0 and X have occurred.

Bayes' rule is to simply choose the minimum risk.

To illustrate the use of Bayes' rule an example will be used that will be encountered in the example section of the dissertation. Suppose that the only classes are A and B. Then the risks involved are

$$r(A) = p(B/0)$$

and

$$r(B) = p(A/0).$$

Then one selects class A if

$$p(B/0) < p(A/0) \qquad 6)$$

and select class B otherwise.

A well known theorem called Bayes' theorem is often used to change the form of the equation. For this problem Bayes' theorem becomes

$$p(B/0) = \frac{p(B)p(0/B)}{p(A)p(0/A) + p(B)p(0/B)} \qquad 7)$$

Substitution into equation 6 gives a new decision rule that one should select class A if

$$p(B)p(0/B) < p(A)p(0/A).$$

Let $p(B) = p(A)$. Then the criterion reduces to simply

$$p(0/B) < p(0/A). \qquad 8)$$

The computation of the a priori conditional probabilities is the usual consequence for problems of this nature. The calculation of these conditional probabilities is all that remains before optimal decisions can be made under the outlined circumstances.

Let the input vector 0 be compared of elements as shown in Figure 4. Furthermore, let class A be generated by selecting p elements from an infinite string of binary elements whose probability of being value 1 is 2/3. The selection is made randomly. Likewise select class B by selecting p elements from a string of binary elements whose probability of being a 1 is 1/6. There is no conditional

probabilities between elements.

A table may be constructed for each value of p listing each of the possible input vectors and the choice that should be made for that input. The percentage right can then be calculated by the formula

$$\% \text{ right} = 100 \sum_{X=A,B} p(X)p(X \text{ is chosen}/X) \qquad 9)$$

where the summation is over all possible classes.

Table 1. Bayes' decision with p=1 (% right = 75%)

0 vector o_1	p(0/A)	p(0/B)	class choice
0	1/3	5/6	B
1	2/3	1/6	A

Table 2. Bayes' decision with p=2 (% right = 79%)

0 vector o_1 o_2	p(0/A)	p(0/B)	class choice
0 0	(1/3)(1/3)	(5/6)(5/6)	B
0 1	(1/3)(2/3)	(5/6)(1/6)	A
1 0	(2/3)(1/3)	(1/6)(5/6)	A
1 1	(2/3)(2/3)	(1/6)(1/6)	A

The application of this rule is possible only for uninteresting inputs. Suppose that the pattern recognition system was that of Figure 2. Then the ideal pattern recognizer will have only one output class for any input 0. What this means is that

$p(0/X) > 0$ for X equal to the correct class

$p(0/X) = 0$ for X other than the correct class,

for a particular input vector 0. This conclusion can be reached the first time that the pattern recognizer observes the input vector 0 in the learning set. If 0 is not explicitly in the learning set, there is no way, without assumptions, to calculate $p(0/X)$, for any X.

Bayes' rule is useful for the pattern recognition system of Figure 1 when the inverse of the pattern generator doesn't exist. In this case 0 may belong to more than one class, such as in the previous example. Even for this case problems exist, for the learning set is almost always insufficient in total number of vectors to generate adequate statistics.

There is one case where the learning set and test set are the same, so that the learning set doesn't have to be finite. In the case of prediction the machine may be utilized even while it is learning. The pattern recognizer must predict the class, but after some delay the correct answer will be available to it. One might wish to use the machine

even while it is learning, since it may be the best predictor one has!

2. Weighting functions

This is a method of pattern recognition which is prevalent throughout the literature (6). Many variations on the basic theme are possible. The technique is closely related to template matching, if not the same.

The input vector is usually a vector defined on Euclidean n space, R^n. A function is defined, perhaps a linear functional, which, when operating on the input vector, reduces the vector to a scalar quantity. A separate function is necessary for each class under consideration. For each input vector of the test set, a scalar is generated for each defined function and hence for each class possibility. The class is selected for which its corresponding function produces the largest scalar. Other comparison criteria are also used.

The learning portion of these schemes involves determining the coefficients of the functions so that errors in the learning set are minimal.

C. Correlation with Stored References

A technique which illustrates the above philosophy is the use of correlation (12). In this case the function which is used is the inner product defined on R^n as

$$P_j = \sum_i x_i y_{ij} \qquad 10)$$

where x_i = i^{th} element of the input vector

y_{ij} = i^{th} coefficient of the reference vector for the class j.

The reference vector for each class is found by computing a mean or average vector from those vectors of that class which are in the learning set. The "closer" that an input vector is to a reference vector, the larger the value of P_j will be. Therefore class j is selected if P_j is the largest among all the possible inner products.

It can be seen that vectors of a given class need to be grouped "near" the reference vector for that class to be correctly identified. A stray vector that is near the reference vector of another class will always be missed. There is no provision in this technique to guarantee that the learning set will be learned.

Another technique that has proved useful is one outlined by Cover and Hart (4). This is known as the nearest neighbor rule. The learning set, or a portion of it, is stored for reference during the test set. For each vector of the test set, the distance from it to each of the stored references is computed. The distance function may be the Euclidean distance, but the precise function seems to be arbitrary as long as it satisfies the usual definition of

distance. The class of the reference vector which is found to be closest to the test vector is taken to be the class of the test vector.

If all the learning set is memorized as references, it is obvious that the entire learning set will be learned. The problem which arises is that a large amount of memory space must be used. An alternative is to memorize only a select few of the learning set to use as references. This process might reduce the amount of memory drastically, and yet reduce the performance an insignificant amount.

IV. THE PREDISTORTION NETWORK

A. Preliminary Concepts

A typical way to design a pattern recognizer is to first examine the problem in the light of a given classifier. The initial one-to-one, preprocessing, and feature extraction transformations are conceived by the person who knows the limitations of the classifier. After these transformations are established, the classifier is used, with the output of the feature extractor as its input, to learn the way to classify the input patterns correctly.

An example will illustrate an important point. Suppose that the input is a series of measurements which will hopefully determine an airplane's location. The classifier must determine if the airplane is within 5 miles ground distance. The two classes are yes, it is close, and no, it is not close. Suppose that the measurements are the actual line of sight distance and the angle of inclination to the airplane. If these two measurements are given to a classifier, it must learn to use trigonometry, or an approximation to it, to calculate the ground distance, and then learn to compare the distance with 5 miles to conclude the correct classification. If the person who designed the feature extractor was clever enough to extract the horizontal distance and use that as an output of the feature extractor,

the classifier would have only a simple comparison to learn. The point is that often times the person does the learning that is difficult, and leaves only the trivial learning for the classifier. The credit for the learning should not go completely to the classifier.

The scope of this dissertation is to describe a preprocessing system. One is initially given a classifier, feature extractor, and preprocessor as shown in Figure 3. It is also assumed that the elements of the 0 vector can take on only the values of 1 or 0. It can be called a binary vector. The classifier is assumed to be fixed, that is, it has previously been taught and no more learning will be done by it. Finally it is assumed that the system given is operating inadequately.

There are several reasons why a pattern recognition system may not be operating satisfactorily. It may be that the feature extractor is not extracting significant information for the classifier. This is apt to happen because the feature extractor was built using incomplete knowledge about how the problem might be solved and how the classifier operates.

It may be that the classifier is inadequate to handle the information that the feature extractor is giving it. For example, perhaps the classifier can separate only those classes that are linearly separable (6).

Another possibility is that the decision boundaries have changed because of nonstationary statistics. One would like to alter the classifier's decision boundaries slightly without undergoing a major revision of the entire system. The internal ordering of the classifier might dictate that this is not practical.

This paper describes a way that a preprocessor might be inserted into the system to improve its performance, without knowing the explicit internal function of the feature extractor or the classifier. The preprocessor will also be called a predistortion network. This terminology is used because the network to be added to the system may be thought of as distorting the image to be received by the feature extractor to a form which is more easily reduced for identification by the classifier.

The power of this approach can be seen if one notes that all that is really necessary to show is that <u>some</u> improvement can be realized by inserting the predistortion network. If the improvement gained by the first predistortion network is not enough, another network may be added to the left of the first one, and additional improvement can be realized. This chain of improvement possibilities is not possible when the feature extractor and preprocessing are fixed first and improvement is sought by adding networks to the right of these. This is because

these networks tend to reduce the information content available to each succeeding network to its right.

The purpose of a predistortion network is two fold. In the aggregate it can be thought of as a process that transforms its input vector to an output vector that is in a better form for proper recognition by the system operating on its output. Its first purpose is to determine what is important in the object vector for making an improvement in the pattern recognizer. Secondly, it must determine how to alter the input to the feature extractor so that this improvement may be realized.

One approach to a solution for improvement is to make an analogy between the person of the airplane example and the predistortion network. That is, have the predistortion network solve the classification by itself, and cause its output be of a form that is very easily recognized by the classifier. This would be an acceptable solution except for the fact that the present feature extractor-classifier may be an extremely complicated piece of hardware that may be clever in its own right. It is more desirable to have the predistortion network compliment and complete the feature extractor-classifier unit. The predistortion network should be like using a better antenna for a television set to improve its reception, rather than like building a whole new television set. Keep in mind, however, that if the

feature extractor-classifier is capable of only the most trivial of decisions, the predistortion network must do a maximal amount of distortion and class separation.

B. The Transfer Function

The process by which the preprocessor "learns" to improve the pattern recognition system can be outlined. The input to the predistortion network is a binary number. That is, each of the elements of the number or vector can take on the value of one or zero only. This vector is represented by the letter "O". Any element of the vector can be represented by o_i. Let O have a total of p elements. The output vector is again a binary vector, denoted by N. A typical element is the n_j element. Let the vector have a total of q elements. The object of the distortion network is to generate a function F that will transform the vector N such that the output of the classifier corresponds to the true class of the input vector O. The transfer function and the O and N vectors are shown in Figure 4.

If a solution to the pattern recognition problem exists, a function can be found to perform the desired transformation. Since the input is essentially a number with a finite number of elements, there are only a finite number of possible values that it can assume. Therefore one has only to associate with each input vector an output vector that will

cause the classifier to classify the vector O correctly. If there are only a finite number of input vectors, each having a finite number of elements, then it is always possible to write a set of Boolean functions to describe any desired output.

Even though the above paragraph points out that a function is possible if a solution exists, the use of a function that is a complete list of all the possible inputs is not a practical approach to the problem solution.

The number of possible input vectors that must be listed may be an astronomical number. If the vector O has n elements, then there are 2^n possible vectors. Secondly, the training set may not contain all the possible input vectors. Only a representative set may be available. Without the benefit of all the possible input vectors, one must learn how to distort vectors that have never been seen before. To do this the pattern recognizer or distortion network must form an hypothesis about what class unseen vectors are a member. This procedure is a learning process.

A close approximation to the function F will be generated by examining a limited number of learning examples and forming a hypothesis to classify vectors not explicitly appearing in the learning set.

There are several points to be made concerning what a desirable solution is. First, a simple problem should have a

simple solution. The effort expended should be at most proportional to the difficulty of the problem.

The distortion network should be designed so that the learning set can be learned. If all the possible vectors were in the learning set, (this would not be known a priori), then a solution would have to mean that all the learning set is learned. Thirdly, the learning of the learning set is only secondary to forming an adequate hypothesis about the classification of vectors in the test set. The best index of performance is how well the pattern recognizer does on the test set, but this index is not available during the training period. As mentioned previously, the network should complement the existing system, rather than duplicate it.

C. The Correlation Function

The function $g_{rs}(o_i,t)$ is a function defined such that

$$n_j=g_{rs}(o_i,t)=o_i\cdot\bar{t}+\bar{o}_i\cdot t \qquad (11)$$

or

(r and s are indices of learning to be defined later.)

$n_j=o_i$ if $t=0$

$n_j=\bar{o}_i$ if $t=1$

where $\bar{o}_i$ means "not o_i" .

The function g_{rs} is the basic operational unit of the transfer function F. By iteratively applying this function

to the elements of the N vector, any transformation can be realized.

Theorem 1:

By iterative application of functions of the form of g_{rs}, any one vector O can be transformed to any vector N.

Proof:

Select any element o_k. If n_1 is to be the same as o_k, let t=0. Let s=1 to denote the iteration number. Then let $n_1=g_{r1}(o_k,0)$. In a similar manner, let $n_j=g_{rj}(o_k,0)$ if n_j is to be the same value as o_k, or let $n_j=g_{rj}(o_k,1)$ if n_j is to be of the opposite value of o_k.

The above theorem indicates that any output vector N is realizable if a certain vector O is known to exist at the input. More generally, there are a large number of O's that will produce the same changes in the output vector. The above theorem is easily applied once an appropriate N vector can be decided. To do this the classification of the input vector O must be known. But this requires a solution to the pattern recognition problem itself! The desired approach is to create a function that produces a distortion in the vector N such that regardless of the class of the input, the output will be recognized correctly.

Each time that the function g_{rs} is applied to the input

vector the input is said to undergo an iteration. The value of s is the iteration number. From the foregoing proof we can see that o_k can be selected in an arbitrary manner to effect the desired output. Therefore one may select the element o_k such that all the input vectors that are operated on by g_{rs} are divided into two sets, those where $o_k=1$, and those where $o_k=0$. On the next iteration two groups will exist, according to the division brought about by o_k. For the first group let r=1, and for the second group let r=5. One may now determine two functions, $g_{1(s+1)}$ and $g_{5(s+1)}$, which will further divide the learning set. By continuing in this manner the learning set may be arbitrarily divided into any number of groups, if enough iterations are used. If b is the number of iterations, then it is possible to divide the learning set into 2^{b-1} different groups. See Figure 5.

Theorem 2:

By using successive iterations and application of the function g_{rs}, on the learning set, it is possible to realize any Boolean transfer function F.

Proof:

If the learning set consists of L vectors, then by the application of at least $\ln_2(L-1)$ but less than L iterations, the learning set can be divided into L groups,

each group having only one vector as a member. This can be done by choosing for each $g_{rs}(o_i,t)$ an o_i that at least one member of the subset of vectors at point r,s does not have. At least one such point can be found unless all the members of the subset are identical. By theorem 1 an additional q iterations (or less), one for each element in the vector N, will be sufficient to give any desired output.

The above discussion shows that any learning set can be learned. That is, any output for any input vector for every member of the learning set. The ability to learn the test set by examining only the learning set is the crux issue. Also the other desirable features of the predistortion network need to be verified.

The learning portion consists of deciding, for each $n_j=g_{rs}(o_i,t)$, values of j,i,t, as well as several bookeeping decisions to be covered later.

Hypothesis 1:

The o_i's should be selected to be significant elements in that if the learning set is divided into groups A and B by point o_i, then the vectors in the test set will also be divided into similar groups A and B.

As an example, group A could be vectors that are of a certain class, and group B could be all other vectors. Or group A could be all the vectors that are presently being

classified incorrectly by the pattern recognizer, and the B group would be those vectors classified correctly.

Hypothesis 2:

The selection of n_j and t should be such that the N vector is changed in a manner that will increase the likelihood that the feature extractor and classifier will correctly differentiate between those groups A and B as designated by the value o_i of Hypothesis 1.

To facilitate the discussion and to clarify the process, one can think of an imaginary particle traveling along a decision pathway for each input vector O. Each pathway divides at each iteration. The imaginary particle follows the fork in the pathway according to whether the input vector possesses the o_i of the g_{rs} function of that decision point. Also at each iteration point the N vector may or may not have n_j altered, according to the value of t on the function g_{rs}. After all of the iterations have been completed, the feature extractor and classifier operate on the amended N vector to determine the class of the N vector and hence of the O vector. Figure 5 shows the decision pathway system. This decision pathway system constitutes the operating memory of the predistortion network, and its Boolean equivalent is an approximation to the desired transfer function F.

D. The Learning Process

The learning procedure is one of determining the variables of the g_{rs} function. To this end rules need to be formulated based upon suitable hypotheses.

Hypothesis 3:

Given the groups A and B into which the vectors being operated upon by $g_{rs}(o_i,t)$, o_i should be selected so that the probability of a vector being placed into the wrong group is a minimum.

Minimize the expression

$$p(\text{error}/o_i) = p(\bar{o}_i/A,r,s)p(A,r,s) + p(o_i/B,r,s)p(B,r,s) \qquad 12)$$

where it is assumed that class A has o_i.

Many times this expression is minimized by choosing an o_i as the element that all the class A's and class B's possess. In this case the probability of error is simply p(B). There is no division of the vectors into groups in this case. A next best rule must be used in these cases, so that the learning set can be further subdivided and eventually learned. To circumvent this possibility of choosing such a commonly occurring point, a rule is established.

Rule 1:

No o_i should be selected that all the vectors to be operated on by g_{rs} possess, unless only one class is present.

If Rule 1 is used, then the probability of an error after selecting a point o_i may be less than the probability of an error before the selection of a point. It has been observed that the point determined by Hypothesis 3 under these conditions tends to be insignificant. Typically this is a point that is possessed by only one vector. Again, points of this nature do not significantly divide the set of vectors into significant sets. The learning of the learning set is considerably delayed. Many additional iterations are necessary. To this end an additional rule is used.

Rule 2:

If the probability of an error after point o_i is selected according to hypothesis 1 is less than the probability of an error by selecting no point, then divide the group so that as many vectors as possible of group A take the desired pathway. That is, select the best intraset feature. If possible, optimize to hypothesis 1 as a second consideration.

Consideration must be given as to what the identities of group A and group B are. It has been found that the most significant points seem to occur when divisions are made

along class lines. One other division considered was dividing the vectors classified correctly from those classified incorrectly.

Rule 3:

Group A will be the class of the first vector located, of the vectors that g_{rs} is to be applied to, that is classified incorrectly.

Rule 4:

The point o_i should be selected so that group A takes the upper pathway as shown in Figure 4 if r is less than $r_{max}/2$ and it should take the lower pathway if r is greater than $r_{max}/2$, where r_{max} is defined to be the largest permitted value of r. A computational rule for determining the next state value of r is

$$k_{s+1} = ((r_{max})(u)+k_s)/2 \qquad k_1=0 \qquad (13)$$

where

r, for each value of s, = $r_s = k_s+1$

and

u = 0 for the upper pathway

u = 1 for the lower pathway.

Figure 6 clarifies this equation. The differentiating between groups in the manner as described by rules 3 and 4 creates a tendency for organization of pathway selection.

This rule has its effect when recombination occurs, which is explained later.

An important point and its consequences needs to be considered. That is, the statistical hypothesis which the foregoing rules have been based upon. It is assumed that each element of the learning set vectors occurs statistically, and that the probability of its occurrence in any one class can be determined by examining the actual vectors of the learning set. It is also assumed that the test set elements occur with the exact same statistical occurrence. For instance, if element o_6 is a "one" in 60% of class A in the learning set, it is assumed that element o_6 is a "one" in 60% of all the vectors of class A in the test set also.

After several iterations, the element o_i is selected on the basis of examining the statistics of only a subset of the original learning set. It is entirely possible, and often probable, that only two or three vectors are in a subgroup! It is difficult to assume that this subgroup is a sufficient sample of the learning set and to further say that its statistics are the same as will occur in the test set. The only true way to adequately make a proper statistical decision on any subgroup is to be sure that the learning set is so large that any subgroup that is examined is a good statistical representation. This is a practical impossi-

bility when one realizes that after many iterations only a few vectors will be present at each decision point rs. To help somewhat, an additional weighting factor is computed. This weighting factor is the probability that an error would result if o_i were chosen, but with all the vectors of that class in the learning set being used in the computation, instead of a subset. The weighting is such that it is most influential when the size of the subgroup is small and the probability of error is high.

The selection of n_j and t needs to be considered.

Rule 5:

The element n_j should be selected so that as many N vectors of class A as possible that are incorrect will be altered to be more like a reference vector for class A. The value of t should also be chosen to accomplish this.

Hypothesis 4:

The use of g_{rs} on vectors that are not of class A will be beneficial, despite the fact that only vectors of class A were considered in Rule 5.

The reference vector is a vector supplied by the feature extractor-classifier. It is an input vector to that system that is known to be identified correctly. This reference vector may be chosen arbitrarily otherwise. It does need to be the same vector for every iteration, and for every vector

of the same class.

Again the problem of statistics enters into the selection of n_j, as it did for o_i. An insignificant n_j is apt to be chosen because the subset of vectors being operated on may not be a true representative set. To help relieve the situation, the index of rule 5 is weighted by the probability that all the N vectors of class A of the learning set that have been previously correctly identified possess the element n_j.

E. The Decision Pathway

Examination of Figure 4 shows that for each iteration s, there are 2^{s-1} possible division points. All of these decision points may not be necessary. To avoid such a large expanse of memory space, it is desirable to put an upper limit on the value of r of the function g_{rs}, Figure 5 shows how an upper limit can be placed on r. The essence of the limiting process is that decision particles, having traveled different decision pathways are at the same value of r and s. This condition is called recombination. The recombination of pathways can only be noted by examining more than one iteration at once. Since the predistortion machine only considers one iteration at a time, independent of the past history of the vector, recombination is not a real phenomenon to it.

If adequate separation and advancement on the learning

set is to be made, r_{max} should be large. However, memory space is wasted because not all possibilities of r and s will occur for a learning problem. Small values of r_{max} mean that recombination occurs frequently, and for each time that this happens, the learning process is set back. Therefore more iterations will be necessary to solve the learning set. The rules that have been given take into account recombination to a certain extent and minimize somewhat the learning setback produced by having subgroups of the learning set, that were once separated, back together.

Rule 6:

Whenever all the vectors of the learning set that are at decision point rs are all correct, these vectors are removed from the learning process.

The hypothesis associated with this rule is that if a decision particle of the test set also arrives at this point, the vector that it is associated with will be correctly identified by the classifier.

The advantage of Rule 6 is that the size of the learning set is diminished when the rule is invoked. This decreases the chance that recombination will occur.

F. Related Literature

The previous section has indicated that the predistortion network does a two fold operation. Only a few features or elements of the O vector are used to make decisions. In this respect a feature selection function is performed. Also a minimal distortion of the output is attempted which is sufficient for correct recognition by the feature extractor-classifier. In this sense the distortion network is not a classifier, except in extreme cases.

The selection of the elements from the O vector has been examined only lightly in the literature. The emphasis of the literature is on the determination of the necessary features before the feature-extractor-classifier is built, rather than the selection of features to control the pre-distortion network.

Lewis (11) examines the feature selection problem by defining a criterion of goodness for each feature under consideration. He assumes that the features to be selected from are measurements made on the pattern, rather than binary elements. Also each measurement is statistically independent of any other measurement. Lewis then defines a criterion of "goodness", which is an empirical relation using the joint and conditional probabilities between a feature and the classes. The criterion was selected from specifica-

tions that one would like a good feature to possess, and he notes that all the desirable specifications are not measurable as a single scalar measurement. His criterion is a compromise function.

Through the use of experiments Lewis showed that by using a combination of features that had the largest sum of "goodness" criteria, the pattern recognizer used performed proportionately well.

Chow (2), Chow and Liu (3) developed a method of class separation for a pattern recognizer that is related to the method used in this work.. Chow considers the input vector to be binary in nature, similar to this dissertation. One then notes that the optimum Bayes decision requires that the conditional probabilities be computed as

$$p_j = p(O/C_j) \quad \text{for} \quad j=1,2,3,\ldots, \text{ n classes} \qquad 14)$$

where

C_j is the name of the class

O is the particular input vector under consideration, and to select the class corresponding to the largest p_j. Now since O contains p elements,

$$p_j = p(o_1,o_2,o_3,\ldots,o_p/C_j) \qquad 15)$$

and this can be expanded to

$$p_j = p(o_1/C_j)p(o_2/o_1,C_j)p(o_3/o_2,o_1,C_j)\ldots. \\ \ldots p(o_p/o_{p-1},o_{p-2},\ldots,o_2,o_1,C_j) \qquad 16)$$

Each of the above factors must be stored for each possible class and input configuration. This is a large number of factors. To reduce the amount of memory needed, Chow makes the approximation that

$$p(o_i/o_1,o_2,\ldots o_{i-1},C_j) = p(o_i/o_{i-1},o_k,C_j) \qquad 17)$$

That is, the correlation between features extends only to the feature's nearest neighbors (i.e. o_{i-1} and o_k). With this limitation, the memory space required for the probability factors is reasonable. The equation forms the basis for the decision process in the predistortion network. Consider the computation of p_j using Equation 17. Let the o_i's be selected from the input vector O such that at least one of the factors on the right side of the expression is zero for all classes except one. In addition, let those points or elements be selected such that for the class whose p_j is nonzero, the conditional probability is a maximum. It can be seen that in the test set one only needs to check to see if the elements $o_1,o_2,\ldots o_i$ exist for the particular input vector. If they do the output of the pattern recognizer is that class associated with that set of input elements. The selection of elements is complicated by the fact that every input vector examined in the test set must be classified. The above remarks indicate the classification of only those vectors which have the same particular subset of ele-

ments as the learning vectors. Some suitable hypothesis must be formulated to include all possible input vectors that may be encountered.

The selection of a hypothesis and a selection criterion has been the aim of the preceding section. This has been done by the construction of a Markov chain decision function.

G. The Computer Program

The above ideas were tested by use of the IBM 360 computer, using Fortran programming. Figure 7 shows a simplified flow chart for the program. Some comments are appropriate for each block.

Block 1:

All the input vectors are read into the machine for both the test and learning sets. Learning is done on only the learning set, however. The vectors are put into binary form if they are not already in that form. Also reference N vectors, for use by the feature extractor-classifier (a subprogram) are read in. These references are samples from the learning set that the feature extractor-classifier can identify correctly.

Block 2:

The initial N vectors are computed. Unless stated otherwise, these will be identical to the O vectors. There

is no restriction on what the initial transformation might be, however.

Block 3:

Each N vector is altered by a g_{rs} function that is associated with that position on the decision pathway that the decision particle for that vector is located. Of course for the very first iteration (actually for s=0), no alterations can occur. Each decision particle assumes a value of r for iteration s+1.

Block 4:

The feature extractor-classifier classifies each of the N vectors.

Block 5:

The number right and wrong in each of the learning set and test set is tallied.

Block 6:

A check is made to see if the learning set is learned. If so, the program is terminated.

Block 7:

An o_i and u are determined for each value of r according to hypothesis 3 and rules 1 to 4. Rather than computing equation 12 directly, a value is computed that is monotoni-

cally increasing as the value of equation 12 monotonically decreases.

G=Gain=Total number of A's at the point r,s that have o_i, minus the total number of B's at the point r,s that have o_i. (18)

The value of u is assigned according to whether class A is to take the upper or lower pathway. G is then weighted by the gain for the entire learning set for that o_i. The o_i is selected which corresponds to the largest weighted G. If a negative G is the best possible, the selection is altered by rule 2.

Block 8:

Using the results of Block 7, a value for n_j and t are determined according to rule 5. For each vector of class A at the point r,s that is incorrect, the feature extractor-classifier is asked to supply a reference. A tally is made of all the elements of these incorrect elements to find how many vectors, if n_j were changed, would be more nearly like the reference vector.

A second tally is made to find, for each n_j, how many of the vectors of class A that have been classified correctly also have n_j like the reference vector.

The n_j is selected which has the greatest product of these two tallies. A value of t is selected so that g_{rs} will

effect the proper change.

Block 9:

The machine advances to the next iteration state by increasing the value of s by one. This means that when Block 3 is reentered, the input vectors will be altered according to the values computed in Block 8, and the decision particles of each one will follow the pathway dictated by the values computed in Block 7 and the next state rules shown in Figure 5.

The learning process continues to iterate in this fashion until the learning set is learned.

V. PROBLEM APPLICATIONS

A wide range of problem applications should be possible using these pattern recognition techniques. The input vector O can be any data that can be represented by a binary vector. The limitations on this are those that usually occur with any problem involving quantization of information. The origin of the O vector is not important nor is any meaning which may be connected with each element o_i. For example, the vector might be the binary conversion of a number of measurements made time sequencially on EKG or EEG. Perhaps the input vector is the binary quantization of a letter written on a grid with a pencil.

The feature extractor-classifier must be defined well enough to identify at least one vector correctly from each class. This vector is needed as a reference vector. This is not a stringent requirement because an input vector can be arbitrarily selected if it will create the right output from the classifier. The reference vector doesn't have to bear any relationship to the vectors in the learning set.

The advantage of having a previously defined and operating feature extractor-classifier is that previous work that has been done on the problem can still be utilized. If the classifier is doing well on the learning set, then very little predistortion needs to be done by the predistortion

network, since it tends to operate on only those classes that contain errors, and it operates on parts of the N vector that need correcting.

The data necessary for the application of the preprocessing network is a learning set, test set, and a given feature-extractor-classifier.

Each vector of the learning set must contain enough information to properly classify it. Since this information is not known for certain beforehand, usually a redundant amount of information is used. The number of vectors in the learning set is not fixed, but needs to be sufficient so that the pattern recognizer can adequately judge how to classify vectors of the test set.

The test set needs to have enough samples so that the pattern recognizer can be tested to see if it has learned the pattern.

The feature extractor-classifier needs to be specified initially. If any learning of the classifier is needed, it is assumed that the classifier has done it's learning by examining the learning set and has completed its learning before the predistortion netwrork is applied. It is also assumed that the classifier is able to classify correctly at least one vector for each class, and that those vectors can be used as reference vectors.

It is also assumed that the classifier is inadequate in

the sense that the learning set has not been learned correctly.

In the following examples, the N vector is initially set identical to the 0 vector, unless specified otherwise.

The only parameter that can be adjusted is the value of r_{max}. This is given a value which is a power of two. Too large a value for r_{max} means that memory space is wasted. If r_{max} is too small a value, then recombination is so severe a penalty that learning is done more by chance than by logic. Examination of the decision pathways is sufficient to reveal large scale recombination, and the test set doesn't need to be examined.

The results of these examples are displayed as graphs which are plots of the number right after each iteration for both the learning set and the test set.

For a practical problem, one would ordinarily first learn the learning set before attempting to classify the test set. Only the number right for the test set after the last iteration is ultimately meaningful. This is because in practice the test set is not examined until after the learning is finished.

Classifying the test set after every iteration is useful in examining what would happen if the learning were terminated before the learning set was completely learned. Also it is important to know whether transformations that are

made on the learning set will also be effective on the test set. It is important to have the assurance that improvements in learning set recognition will result in better recognition of the test set, since only the learning set can normally be examined during learning.

The performance of a pattern recognizer is usually measured in terms of the percentage right in the test set. The predistortion network attempts to improve this percentage by improving the percentage right in the learning set over the initial conditions. A performance index for the predistortion network can be defined as

P.I. = 100 - % overall improvement in the learning set + % overall improvement in the test set. 19)

Ideally, the improvement in the learning set should be reflected proportionately in the test set, and the P.I. = 100. It is worthwhile to point out that an operation which improves the performance in the learning set should also improve the performance in the test set. Hence the P.I. is 100. Likewise an operation that decreases the performance of the pattern recognizer in the learning set should reflect that performance in the test set. Again the P.I. would be 100. Regardless of the intermediate steps, if the P.I. remains at 100, the test set will be ultimately learned along with the learning set.

Values of the P.I. that are either above or below 100

indicate that the two sets are not statistically the same in terms of the probabilities measured during learning.

A. Example 1

As was explained in the section on the literature search, Bayes' decision rule gives the optimum decision for cases in which insufficient information is available in the O vector to make a very certain decision. The problem example was selected because of the ease in finding a solution according to the Bayesean decision theory, and hence check the system for one case.

The input vector consists of 8 binary bits. There are two classes. For the first class each bit has a probability of occurrence of 2/3. The probability is independent of any other bit in the vector. For the second class the probability of occurrence of each bit is 1/6. An example of some of the vectors of each class used in the learning set is shown in Figure 8.

A total of 92 examples of each class was split equally between the test and learn sets. Therefore each class occurred equally often. The examples were generated by rolling dice.

The N vector consisted of only one bit and the feature extractor examined only this one bit. This simplification in the possible states of the N vector was made because

this problem is an investigation into the ability of the system to correctly divide the learning set, and not a test of the distortion properties. In this case the predistortion network is essentially a pattern recognizer. It performs essentially as a feature extractor-classifier.

Each iteration consists of examining one more element, and basing the final decision upon that point and the points which were examined during the previous iterations. If only one iteration were allowed, the final decision must be based on only one element. If n iterations were allowed, one could say that n points were examined for the decision and no more.

The example used to describe Bayes' decision rule fits this example. The results for the decision procedure for 1 and 2 points are illustrated in Tables 1 and 2. One can use these results as a standard for examining the results of the predistortion network when it is used as a classifier.

The results of the calculation and of the experimental run are shown in Figure 9. The Bayes' decision rule results are those that would be expected if the test set was operated on after each iteration.

The graph shows that the learning set tends to be learned better than is theoretically possible, and the test set is learned less well than the theory predicts could be possible. The explanation lies in the fact that the

learning set has a finite number of vectors. The training procedure allows decisions to be based on any apparent correlation between elements that seems to be valid. Since only a finite number of vectors exist in the learning set, correlations between elements exist that were not truly intended by the object generator of Figure 2. It is because these false relationships are used by the system that the learning set is able to be learned. Since the vectors in the test set do not possess these extraneous correlations, errors are made, resulting in an error rate that is lower than the Bayes' decision rule. It should be remembered that the Bayes' criterion could be formulated only after the statistics of the problem were enumerated. The pattern recognizer had to learn them from an inadequate (finite) learning set.

After the 4th iteration the number of correct answers in the learning set decreases drastically. This is the result of the fact that changing only one point in the N vector has an extreme effect on the feature extractor used. If the point selected as the o_i doesn't divide the set of vectors in the exact manner hoped for, that is by dividing one class from all other classes, some vectors of class A will be distorted away from class A, and some vectors of class B will be distorted toward class A. Both cases result in incorrect answers appearing when the feature extractor

operates on a minimal number of points. More will be said of this in Example 3.

Figure 10 shows the performance index after each iteration. As the learning set is learned at the expense of using only an approximation to the true statistical distribution intended by the object generator, the learning set is learned without a corresponding increase in the test set. The P.I. indicates this by gradually decreasing as the number of iterations increase. This negative average slope is characteristic of all of the performance index graphs. It is indicative of not learning the correct distortion to be used. It does not mean total failure of the learning, but learning that is less than the complete solution to the problem.

B. Example 2

The O vector for this problem consists of five measurements made on EKG samples. The data was obtained from work done by Brockman (1). The five measurements are shown in Figure 11. The range of the data is shown in Table 3.

The O vector consisted of 35 elements, seven binary bits for each measurement. The lowest value of each measurement was assigned the value 0, and no measurement was then greater than 2^7-1 or 127. Each number was con-

verted to a base 2 number with no regard as to what the number actually represented in the EKG pattern. A sample vector is shown in Table 4.

The two classes are the normal EKG recording and that recording produced by a heart which has a bundle branch block. The description and causes of this phenomenon are well covered in the literature (18). For the purposes here it is sufficient to say that it is detectable from the EKG recording and that it is clinically observed as an increase in the duration of the QRS wave. The QRS measurement increases in the pathological case. The purpose of this experiment is to see if the predistortion network will generate a solution by examination of the QRS internal as is done clinically.

A total of 36 examples of each class were split evenly between the test and learning sets. The N vector was initially set to be identical to the O vector. The feature extractor-classifier was a simple one that examined only the first element of the N vector. This element corresponds to the least significant bit in the atrial frequency measurement. The results of the problem are shown in Figure 12.

The decision pathway for this problem is shown in Figure 13. The values of o_i in the decision pathways show that the QRS is the most important information in the O vector and that the other information can be largely

Table 3. EKG data range

Feature	Range	Quantization	Typical Vector
Atrial Rate	40-150 beats/min.	1 beat/min.	71 beats/min.
Ventricular Rate	40-150 beats/min.	1 beat/min.	71 beats/min.
P-R Interval	0.13-0.24 sec.	0.01 sec.	0.16 sec.
QRS Interval	0.06-0.14 sec.	0.01 sec.	0.07 sec.
Q-T Interval	0.26-0.44 sec.	0.01 sec.	0.36 sec.

Table 4. Typical vector converted to O vector

	A.R.	V.R.	P-R	QRS	Q-T
Binary vector (least significant bit to the left)	1111100	1111100	1100000	1000000	0101000
Element number	01.....	08.....	15.....	22.....	29.....

ignored. It is noteworthy to notice that in the third iteration the pattern recognizer used information in the PR wave measurement. This can be interpreted in two ways. It is possible that the PR wave contains information that is valid when considered with the QRS measurement, and that clinicians have been overlooking this information. This is unlikely in the light of the fact that the solution to this problem seems to be well documented. The other possibility is that the pattern recognizer is picking out noisy information in an effort to learn the learning set. This is usually the result of having only a small number of training examples. This is the most likely situation in this case.

Another run was made but with the QRS measurement set equal to zero in all 0 vectors. The object of this run was to see if the pattern recognizer could detect a pattern without the use of the QRS measurement. The results, as shown in Figure 14, indicate that no large advancement was made in the test set, even after many iterations. It is interesting to note that if none of the available measurements were significant in determining the pathological case, one would expect that an optimum decision of only 50% instead of the 61% obtained. This percentage is most likely a quirk of this data set, but one would have to try many representative learn and test sets and see what the average

results are to be sure.

The performance indexes for the EKG examples are shown in Figure 15. This graph clearly shows that the index drops toward zero when information is used to learn the learning set that is not present in the test set. Note the wide fluctuations about the normal value of 100 for the case where invalid information is used to learn the learning set. This occurs as a result of the fact that the test set doesn't particuarly react to a transformation in the same manner as the learning set.

After the last iteration the P.I. is just the percent improvement in the test set over the original conditions. In this example the final value is 36, rather than a value of 19 if random guesses were made, or a value of zero if no alterations were made in the N vector by the predistortion network.

C. Example 3

For this problem a number, 1,2,3, or 4, was written in a square that had been subdivided into a 6 by 6 grid. If the pencil line of a number passed through a grid square, that grid element was assigned the value of one. If not, the grid element was assigned the value of zero. In this manner an O vector was constructed of 36 elements for each sample number. The first 20 0 vectors of the learning set are

shown in Figure 16.

The learning set consisted of 40 examples, consisting of an equal number of each of the 4 classes. The test set also consisted of the same number of each class.

The investigation using this set of data centers around the effect of using different feature extractor-classifiers. The feature extractor-classifiers are listed in Figure 17. The feature extractors were selected to show varying degrees of performance of successful feature extractors. The selection was made by trial and error methods.

Four reference vectors were chosen from the learning set on the basis that they were classified correctly without any type of transformation. They were chosen on no other basis.

Figures 18, 19, and 20, show the results using the different feature extractors. Figure 18 is the result of using a feature-extractor that is right only 20% of the time! One could improve performance by guessing randomly! The final result is that the pattern recognizer is able to recognize 83% of the test set. Notice the near monotonicity of the learning graph. Notice also how changes in the learning set are reflected in the test set.

Figure 19 is the result of using a slightly better feature extractor. Notice that the curve is anything but monotonic. There is no problem with nonmonotonicity except

that one of the requirements of the predistortion network was that it complement the existing feature extractor-classifier, rather than "unlearning" what had already been accomplished.

Figure 20 demonstrates the same difficulty. Immediately after the first iteration, and up to about the 8^{th} iteration, the pattern recognizer is notable to classify the learning set as well as when learning commenced. It appears that the predistortion network is ignoring previous work that has been done on the problem.

One reason for the wide variations may be that the problem does not have single points that will improve the number right in the learning set, so a temporary set back may occur in order to establish two or more element relationships.

Another cause might be the particular feature extractor used. It was noted in the discussion of the computer program that selection of the o_i for a given r and s is primarily dependent upon the parameter G. The maximum value of G is the number of vectors in the A class and the minimum value is the negative of the number of vectors in the B class.

When G is a maximum, the distortion will be "best" in the sense that the vectors of class A will be "distorted" toward the A reference, and members of other classes will be moved away from the A reference.

When G is less than the maximum value, some vectors will be erroneously distorted toward or away from a reference vector of a different class. When feature extractors are used that are easily influenced by changing only one bit, many errors may result that must be corrected in future iterations.

One concludes that it may be best to distort only when G has a sufficiently large value to insure that the distortion will be appropriate. This consideration will probably only be necessary when the feature extractor-classifier is especially sensitive to single element values, as is true in these examples.

If the number of right answers in the learning set is not nearly monotonically increasing from iteration to iteration, one might apply some rule to limit poor distortions. The observation of monotonicity involves only the learning set, and it is then assumed that the test set is similar.

The following rule is used to give a threshold for when the N vector may be distorted. It takes into account the idea that if the feature extractor-classifier is not identifying many patterns correctly, the distortion should not be restricted as severely than if the feature extractor-classifier is doing well.

Rule 7:

No distortion should be allowed for a value of r and s if the value G selected at that point is less than its maximum possible value times the percentage of the learning set that is correctly identified before that iteration.

It is probably best to examine the learning curve of the learning set to see if Rule 7 should be applied rather than to use it indiscriminately, since this may mean that more iterations will be necessary to produce a distortion.

It is evident from Figure 20 that it might be beneficial to use Rule 7 for this feature extractor. The results of the application of Rule 7 are shown in Figure 21. The learning curve is much more monotonic than before, and fewer iterations are used.

The performance indices are shown in Figure 22 for this example. Notice that the final performance index for the case with the best feature extractor is not as good as those cases where poor extractors were used. This is expected to a certain extent because a good feature extractor makes errors only on a typical vector that occur so infrequently that they cannot be statistically identified using a small sample set.

Figure 23 shows the distortion that occurs for two of the learning set vectors. Notice that the distortion is concentrated in the elements that are observed by the feature

extractor. It is not possible to interpret these distortions in a geometric fashion.

D. Example 4

This problem is the first of two problems which have the same set of object vectors but for which the decision boundaries have been varied. The object vectors consist of the binary equivalent of the paired numbers x and y, where x and y each take on all integer values from 0 to 12. Each object vector O has eight elements, four for each number. There are a total of 169 different possible vectors that can be generated. These vectors can be conveniently represented on a Cartesian coordinate system where the x and y are the abscissa and ordinate. For each problem the 169 vectors are randomly divided between the test and learn sets. The feature extractor-classifier is easily specified on the computer by first transforming the N vector back into it's equivalent decimal numbers x and y, and then using ordinary arithmetic functions to determine where the vector lies in relation to the specified decision boundary. As in the previous problems, the N vector is set initially identical to the O input vector.

This set of problems present a difficult vector space for the distortion network. First, if an element is changed in the N vector, the vector may be transformed to a position

that may be adjacent to its previous location or far from it. The change may be trivial or drastic, depending on the significance of the element. Any class boundary that is not horizontal or vertical is difficult to describe by the use of Boolean functions. This means that many iterations may be necessary to adequately describe the boundary, if it is known. With only 85 vectors in the learning set, the boundary between classes is not adequately described. There is one and only one possible solution to the problem, rather than in the case where the classes are separated geometrically to such an extent that any of a number of decision boundaries are adequate to give correct recognition.

For example 4 the decision boundary for the 3 classes is shown in Figure 24 and three feature extractors are listed. The first extractor-classifier approximates the true decision boundary rather poorly, while the third feature extractor-classifier is a very good approximation to the true decision boundary. It was found that $r_{max}=8$ was sufficient.

The learning curve for the first feature extractor was sufficiently monotonic so that Rule 7 was not used. For the case of Figure 26, it was found that the learning curve dropped on the first iteration to only 34% right from an initial value of 64%. This is sufficiently nonmonotonic to warrant the use of Rule 7.

For the case of Figure 27, the percent correct dropped

to only 34% on second iteration after being at 88% after the first iteration. Again this warrants the use of Rule 7.

The results for feature extractor 1 and 2 seem satisfactory and need little explanation. The results for the 3rd extractor show that it did worse on the test set than before the predistortion was applied. Little can be said in the defense of a situation of this type, but excuses abound. A slightly different learning set might have shown an overall improvement in the results. The loss is small, statistically, and is offset somewhat by the fact that the learning set was learned.

The P.I. for this example is shown in Figure 28, and is typical of that expected. Notice the negative value that resulted for Figure 27.

E. Example 5

Example 5 used the same input vectors but the decision boundary was changed to that shown in Figure 29. The feature extractors used are also shown in that figure.

The first feature extractor is simply a vertical line. This is certainly an unappropriate boundary for this problem, since the reader knows the correct boundary line is a circle. As a "first guess" on a problem it is not unusual to attempt this type of separation. As can be seen in Figure 30, it was decided that the learning set curve was sufficiently mono-

tonic so Rule 7 was not used. The errors that remained in the test set after learning was completed were scattered about on the coordinate grid, rather then being grouped into a localized geometric area.

For the case of the second feature extractor, Rule 7 was used because the percentage right in the learning set dropped below the initial 84% after the first iteration and did not return until after the ninth iteration. As is shown in Figure 31, no gain was made in the test set, but the learning set was learned. To learn the test set an approximation to the inner circle of Figure 29 would have to have been made. Examination of the results show that of the 13 test vectors in the inner circle, two were correctly identified as class 1, while two vectors just outside the inner circle were incorrectly identified as class 1.

The performance indexes for example 5 are shown in Figure 32. The P.I. for Figure 31 ends at zero because no improvement was shown in the test set.

F. Example 6

In the previous examples the entire learning set was held in memory while learning was done. It seems quite likely that for a problem with a very large learning set that the computer memory would not be large enough. It would be desirable, then, to learn on only a portion of the learn-

ing set, then set that subset aside and take another subset of the learning set to further improve the pattern recognizer. New subsets would be used until the learning set is exhausted.

To accomplish this one notes that the N vector was initially set equal to the O vector in the previous examples, but that any transformation is permissible. Using a predistortion network is just such a transformation.

The learning of the first subset, denoted by S_1, is done in a manner identical to the previous examples. After learning is completed, the net result is a transformation from the O vector to the N vector, denoted by F_1. Now the second subset is taken to be learned, but first F_1 is used to initially distort the N vector. In the previous examples one noted that the test set is almost always improved for any learning set. Therefore one would expect the test set to improve after learning on S_2. Now additional subsets can be taken, with all the results of the learning on previous subsets used to distort the O vector.

Example 6 is identical to Example 1 in its format. The exact same test and learn sets were used. The only difference is that Rule 7 is used, and example 6 is done as suggested in the previous paragraph, by dividing the first 88 vectors of the learning set into 4 equal parts of 22 vectors each. Each of these parts was used as a subset. These were iteratively used to produce F_1, F_2, F_3, and F_4, as outlined.

Figure 33 shows the results of this problem. After each subset, an iteration was used just to set up the next subset. This means that only 18 iterations were used in the learning. In example 1 an analogy was drawn between the results and those expected using Bayes' rule. The analogy holds for each subset considered as a separate problem, but not for the entire run as a whole. Also Rule 7 may delay the results of an iteration from appearing until a future iteration.

Since 3/4ths of the learning set is used in a manner similar to the test set, one expects it to follow the test set, but it is expected that it will be classified better than the test set.

The figure verifies to a good degree that the subset needs to approximate the statistical data only to a general degree, and that future learning tends to build on that done before.

The performance index between the entire learning set and the test set could not be much better. The index is shown in Figure 34.

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER INVESTIGATION

A. Conclusions

This dissertation is a different approach to pattern recognition. Previous approaches have been classifiers, whereby any feature extraction or preprocessing is done prior to the learning by the classifier. Often the ability of the classifier is limited or enhanced by the form of the feature extraction operation.

In this work it is assumed that the classifier is fixed, and that learning will be done by distorting the input to the classifier so that the number of correct classifications in the learning set will be maximized and errors in the test set will not be increased. Rather than attempting to learn all of the decision boundaries of the problem, an attempt is made to learn only those portions of the boundary that have not been learned by the classifier.

This approach may be considered more general than previous approaches. The predistortion network is equivalent to a classifier if the N vector contains only $\log_2$(number of classes) elements. This means that the predistortion network must do all of the learning. This was the case in Example 1. In the general case the network does only a portion of the learning.

The predistortion network is constructed by the use of a learning process. This process is a Markovian decision process based on feature relationships which divides and subdivides the learning set by classes. The process of dividing the learning set was defined so that members of the test set would be similarly divided. A correlation function was defined to distort the N vector so that all the members of the learning set can be ultimately learned. Several additional rules were used so that learning would proceed smoothly and with direction.

The examples given justify stating that the use of the predistortion network will tend to be beneficial whenever a pattern recognition system is defined that is inadequate in the sense that it is unable to learn the learning set. The addition of the network will always improve the system in the sense that the learning set will be learned correctly. Furthermore the examples tend to show that the classifier with the network added will do better when classifying the test set.

There appears to be no limitation on the kind of input, except that each element of the O vector must be interpreted as a feature measurement.

This network is also well suited to problems for which partial solutions have been previously determined. Any known information concerning how the problem should be

solved can be incorporated into the feature extractor-classifier. The predistortion network can then be applied to the input of the feature extractor to complete the learning of the learning set.

Use of the network is valuable when little is known about the problem solution. In this case a large redundant set of measurements is taken on the object and used to construct the O vector. After the network has learned the learning set, the measurements that were not used by the learning network can be omitted when the test set is classified. This reduces the amount of effort required for the data acquisition, as well as a reduction in memory space.

Several considerations were mentioned that are desirable for a predistortion network. It is desirable that if a network is inserted into an existing system, the network should improve the system. The examples show that improvement possibilities are best when the feature extractor-classifier is doing poorly, and not as good when the feature extractor-classifier is making errors only on input vectors which are not "representative" of most vectors in their class. The distortion network is still valuable in these cases because it will learn these vectors if they appear in the learning set.

It was pointed out that simple problems should have simple solutions. One might denote the complexity of a

solution by

$$\text{Complexity} = C = (r_{max})(s_{max}) \qquad 19)$$

Where s_{max} is the total number of iterations used.

This is an approximation to the memory used. Example 2 illustrates how little memory might be needed when the QRS interval is a measurement (Figure 12). For this case, C=16. Much more memory is needed when the problem is attempted with inadequate information, such as when the QRS interval is not measured (Figure 14). For this case, C=48.

The problems illustrated were small with respect to what might be encounted. Increasing the size of the input vector increases the computer time required for learning by a linear factor. The memory space required is increased by a factor proportional to the logarithm of the increase in the input vector.

An increase in the size of the data set will tend to linearly increase the computer time required for learning. Example 6 required only a little more time than Example 1.

If the problems are more complex, r_{max} must be increased. This doesn't increase the computation time, but it increases the memory by a linear factor proportional to r_{max}.

B. Recommendations for Further Investigation

Some investigation might be made into the possibility of having a maximum for the number of possible iterations. In these problems it has been assumed that the number of iterations may be extended indefinitely. If the number of iterations is large, it may be possible to drop the first few iterations and not alter the recognition ability significantly. In this way an adaptive provision for a limited memory is possible.

An interesting problem is an extension to the situation whereby more than one reference vector is permitted per class. In this work the restriction that each class have only one reference vector was made so that complete learning of the learning set would be possible. That is, if necessary, every vector of a given class would be made identical to the reference vector. Suppose that the feature extractor-classifier is similar to an environment in that the N vector is an operation performed on the environment, and the output of the classifier is either the class "good" or "bad". The only correct class is "good". For each vector that is classified "bad", the environment must suggest an N vector that would be classified "good", for the particular O vector given. A wide variety of reference vectors might be given despite the fact that all incorrect classifications are of

the same class. An analogy can be made in that the O vector can be thought of as perceptive inputs of the environment, and the N vector would be motor action on the environment.

A profitable case for study would be how to handle inputs that are known to be measurements in a base system other than the base 2. That is, how might a measurement be handled that is in the base 10 system without converting it to the base 2 system? A possibility might be to let the parameter t take on any of the possible values of the base being used. Then decision particles would take the alternate pathways according to whether the measurement is greater or less than the value of t. Appropriate learning rules would have to be devised.

Example 6 illustrated how learning might be done in parts, by examining only a part of the learning set at a time and then building a hierarchy of F functions, each an improvement on the previous ones. Another way to build a hierarchy would be to suppress any previous predistortion networks into the feature extractor. That is, after the first subset is learned, a new problem is begun using the second subset, but now what was the old O vector is designated as the second problem's N vector. This should be a powerful method, because the feature extractor will become very proficient and should be well matched to additional predistortion networks.

VII. FIGURES

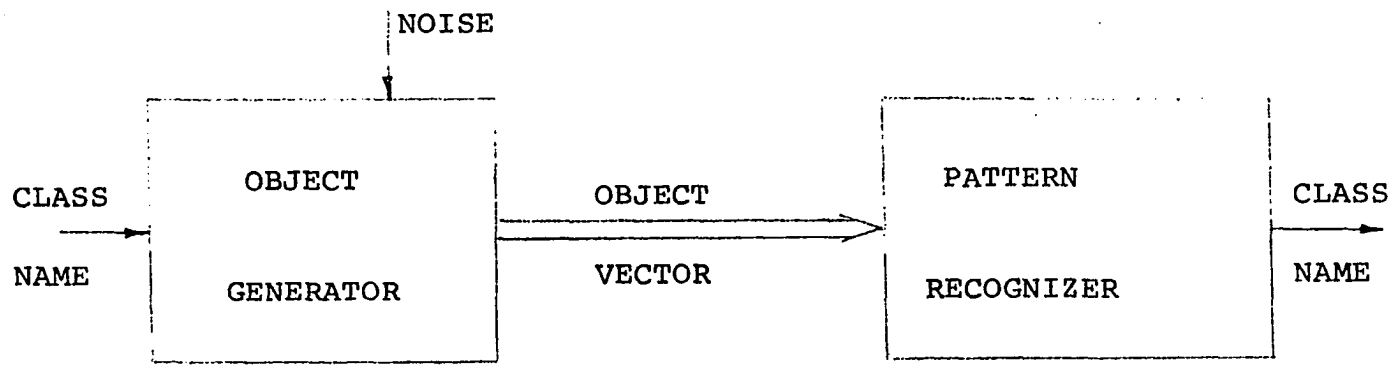


Figure 1. Object generator system

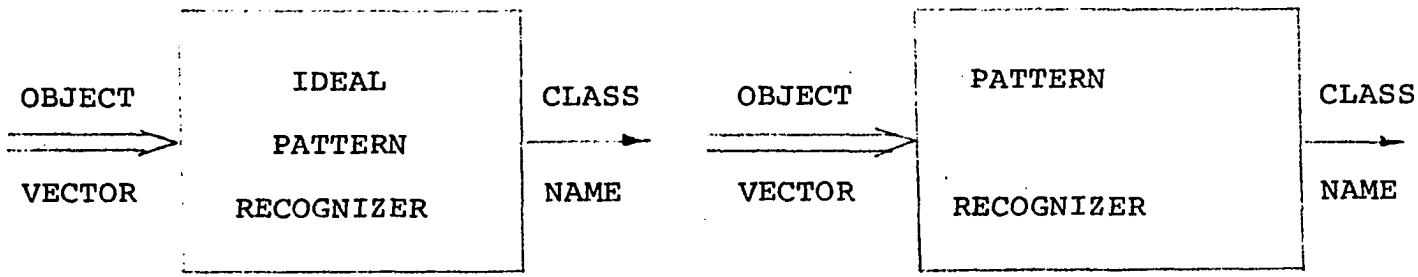


Figure 2. Ideal pattern recognizer system

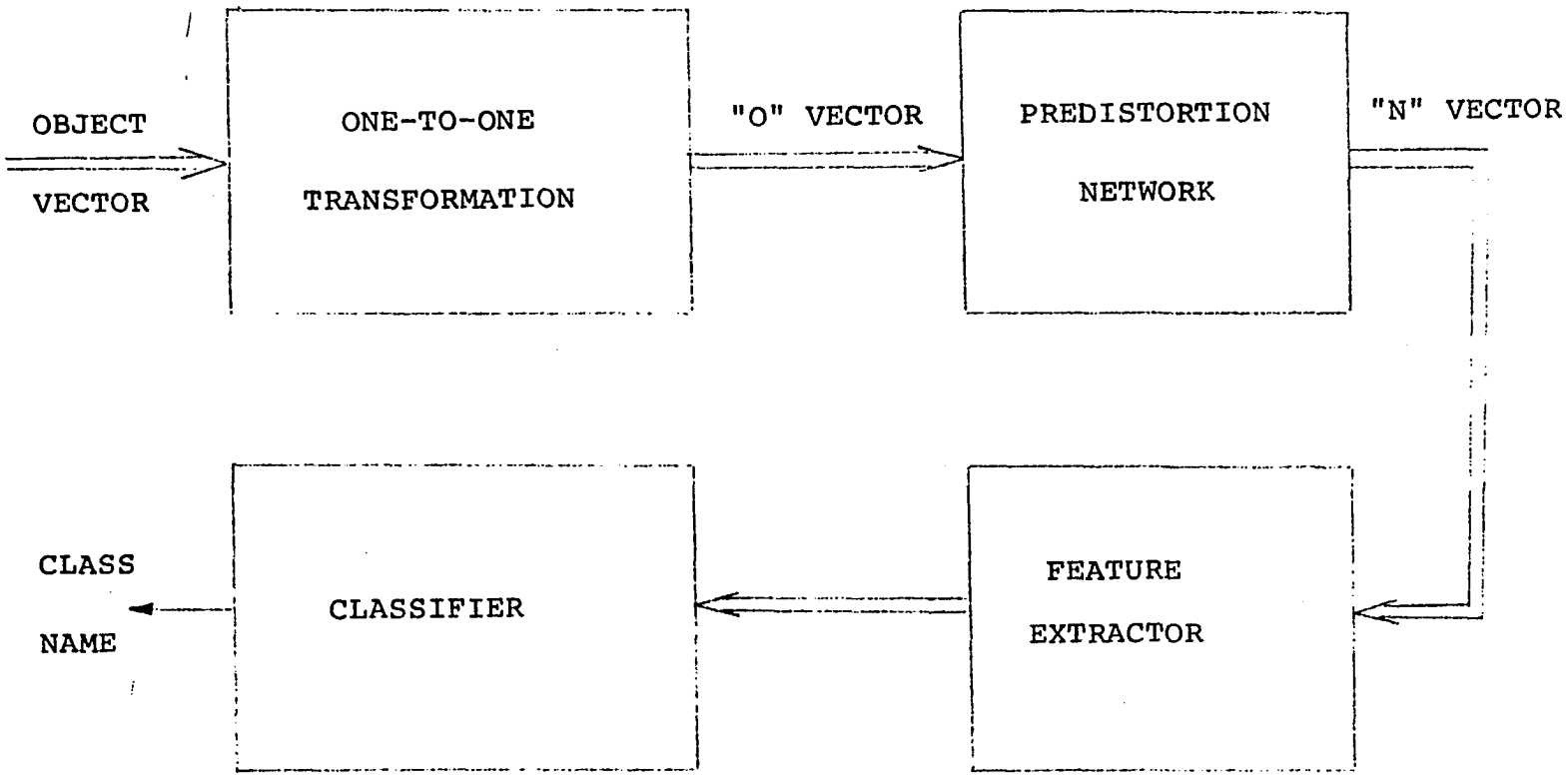


Figure 3. Pattern recognizer

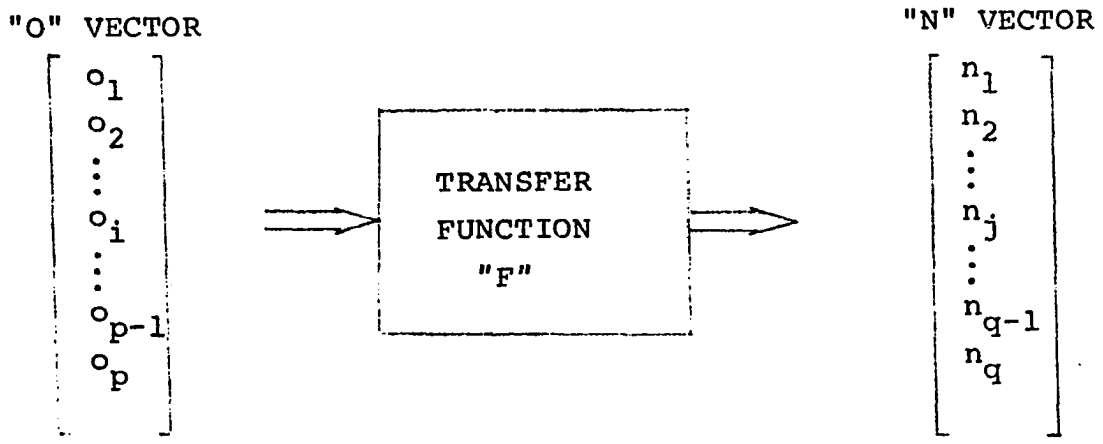


Figure 4. Predistortion transfer function

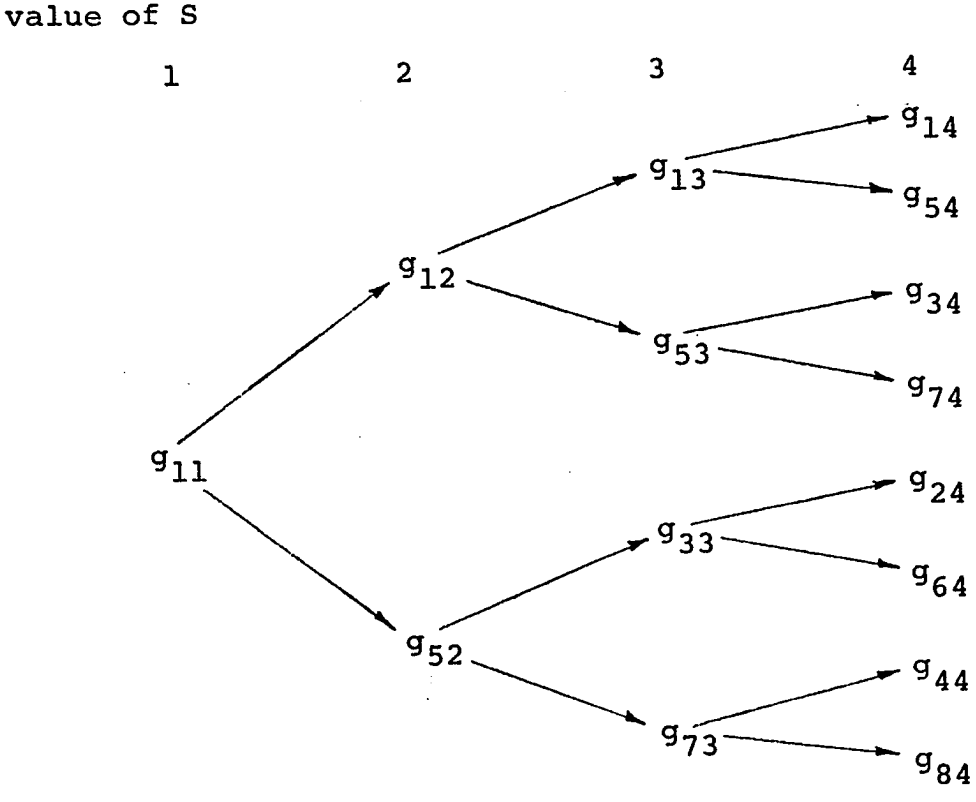
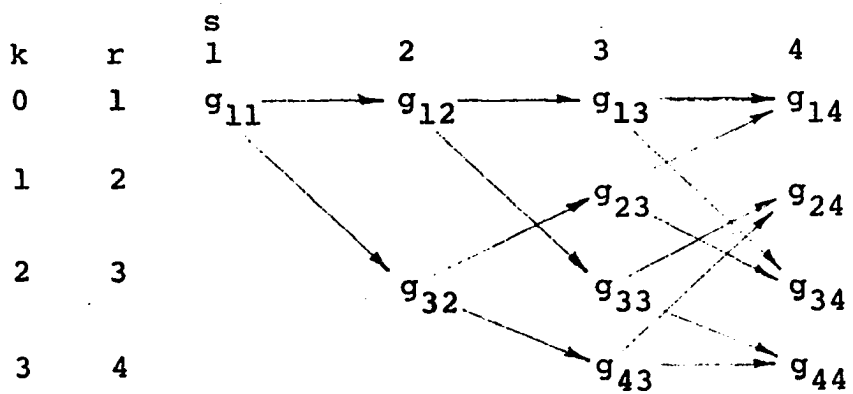


Figure 5. A decision pathway system



Next state formulas

1. Upper pathway

$$k_{s+1} = k_s/2$$

2. Lower pathway

$$k_{s+1} = (r_{max}+k_s)/2$$

Figure 6. Decision pathways for $r_{max}=4$

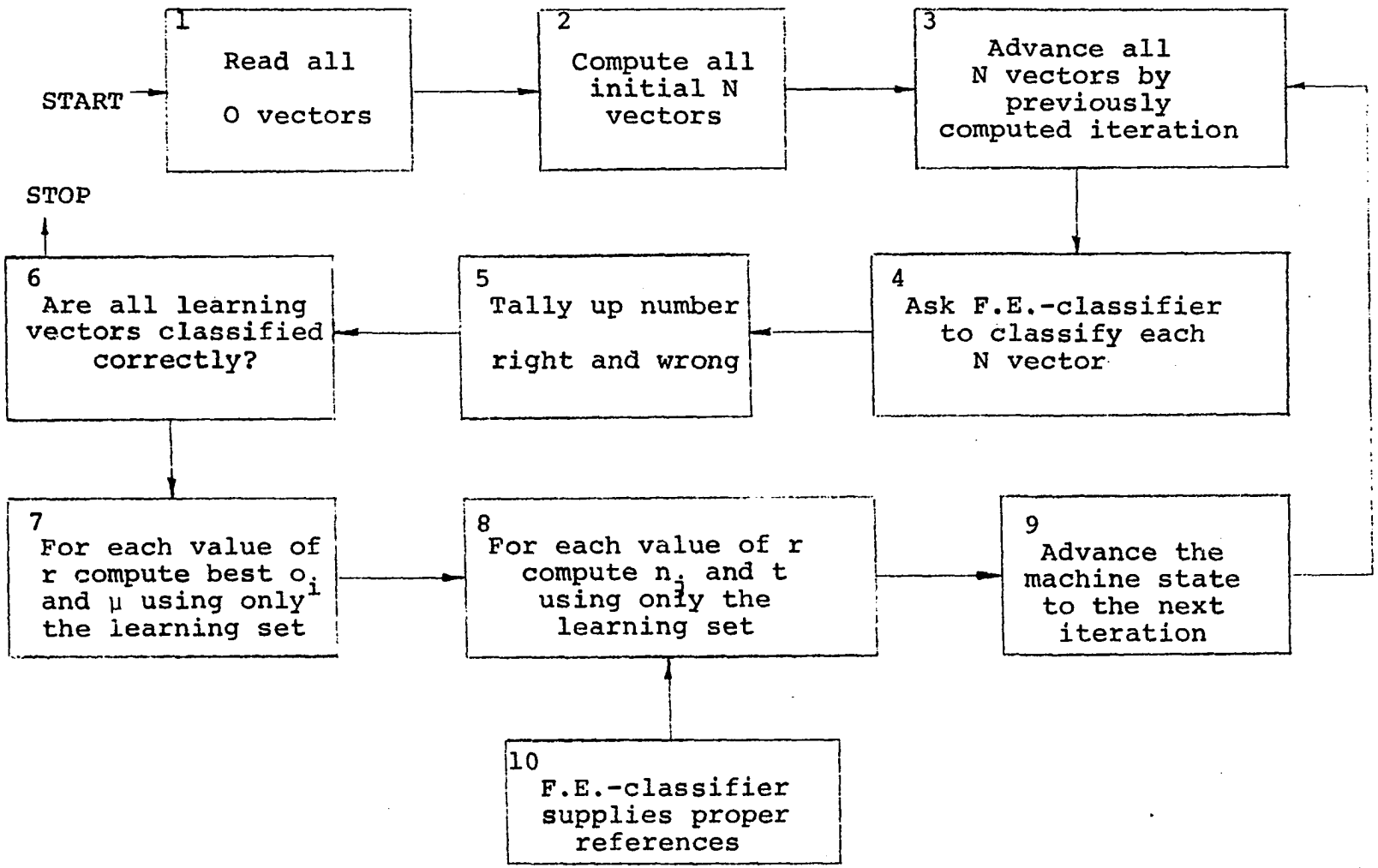


Figure 7. Computer program

CLASS A	CLASS B
1 0 1 0 1 1 1 1	0 1 0 0 0 1 0 0
1 1 0 1 1 1 1 1	0 0 1 0 0 0 0 0
1 1 1 1 1 1 0 0	1 0 0 0 1 0 0 0
1 1 1 1 0 1 1 0	0 0 0 1 0 0 0 0
1 1 1 1 1 0 1 0	0 1 0 0 0 1 0 1
1 1 1 1 0 1 1 0	1 0 0 0 0 0 0 1
1 1 1 1 0 1 1 1	0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0	1 0 0 0 0 0 0 0
1 1 1 0 1 1 1 1	0 0 0 0 0 1 0 1
1 1 0 1 1 0 0 0	0 0 0 0 0 1 0 1
1 0 1 0 1 0 1 0	1 0 0 0 1 0 0 0
1 0 1 1 1 0 1 1	0 0 0 1 0 0 0 1

Figure 8. Sample vectors from example 1

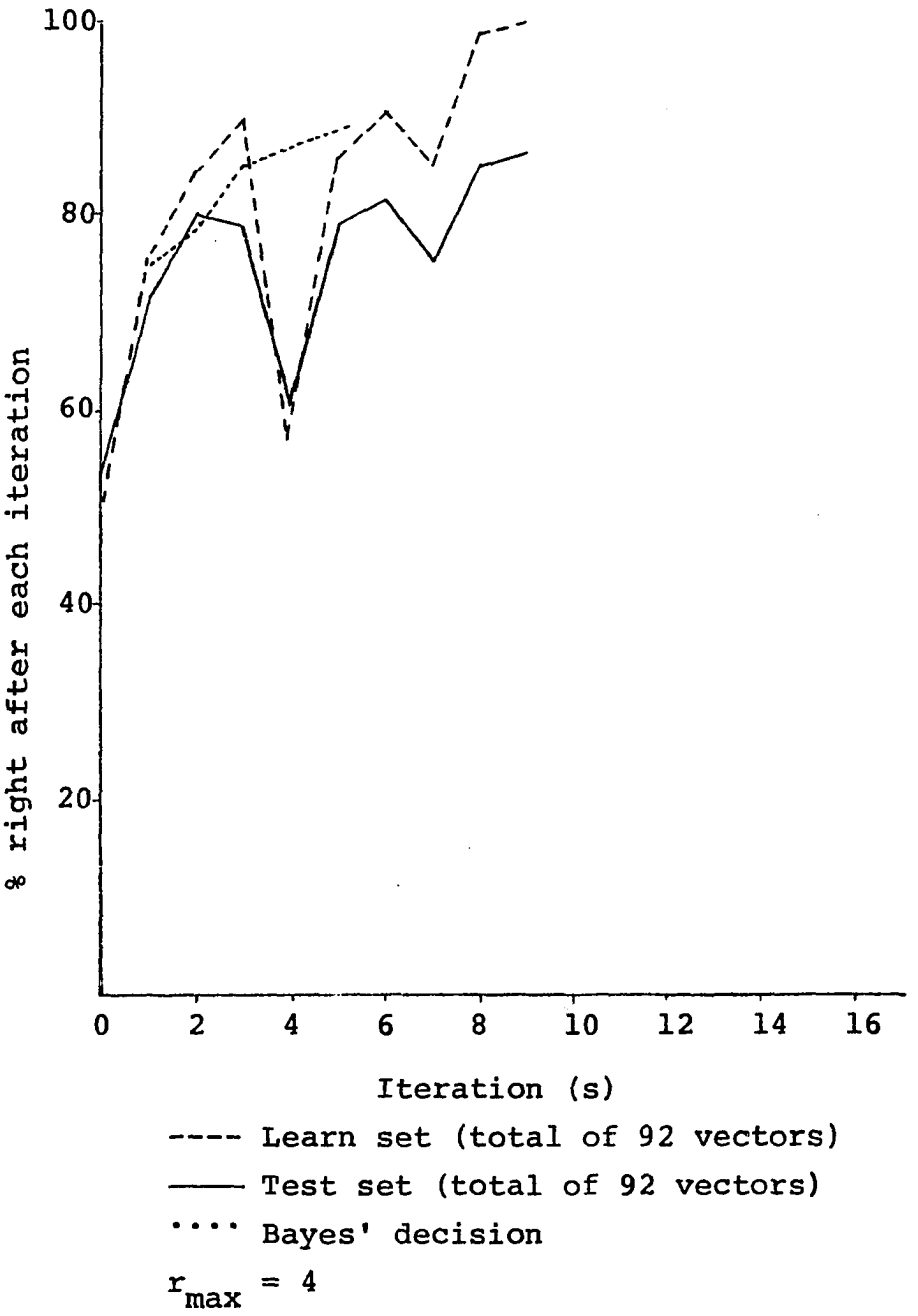


Figure 9. Example 1. Independent features

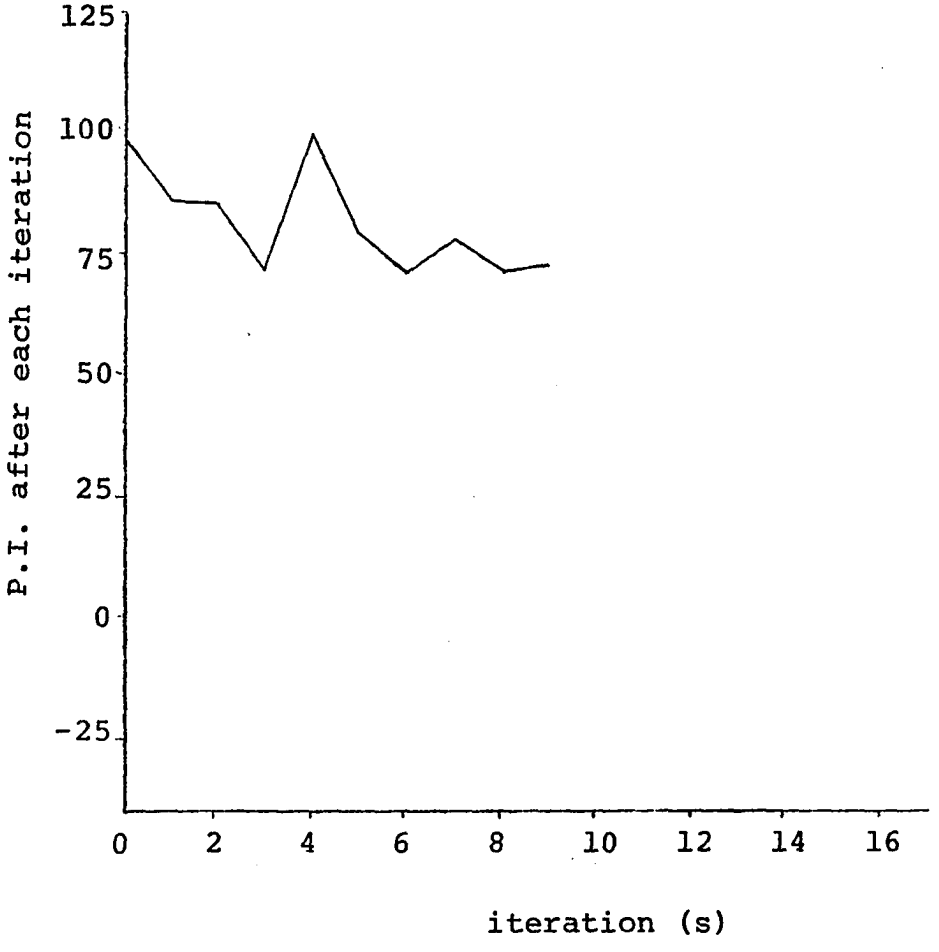
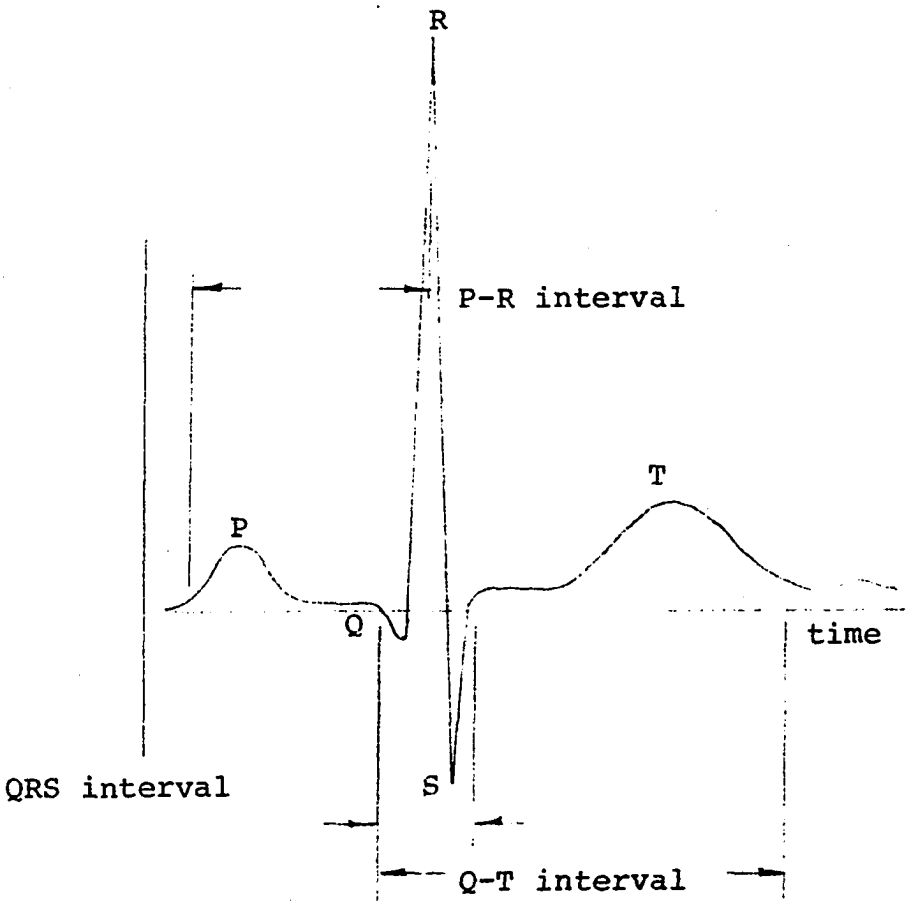


Figure 10. Performance index for Example 1



Measurements

1. Atrial beat frequency
2. Ventricular beat frequency
3. P-R interval
4. QRS interval
5. Q-T interval

Figure 11. The normal EKG and measurements

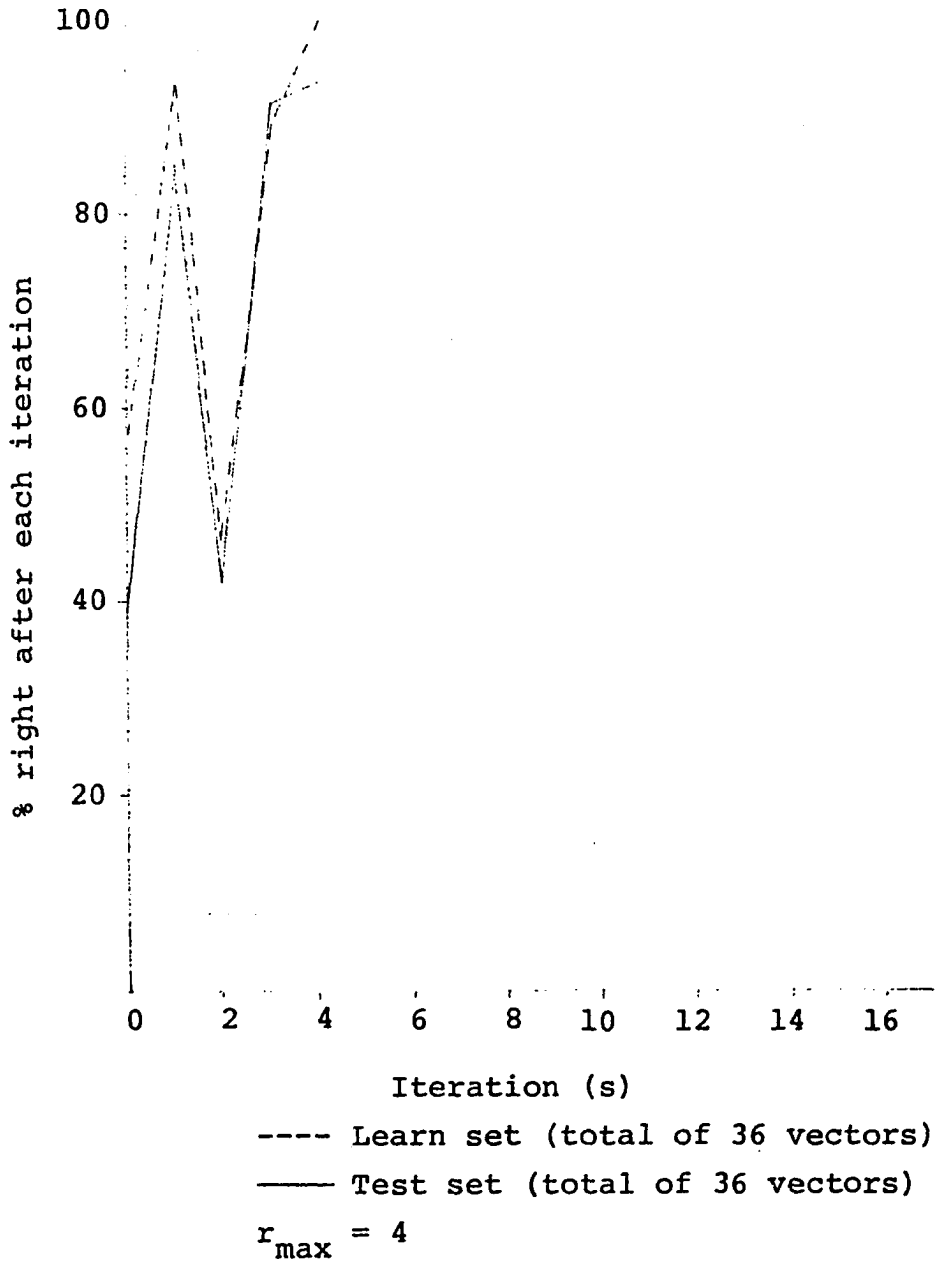


Figure 12. Example 2. EKG measurements

Let (j,i,t),u represent $n_j=g_{rs}(o_i,t)$ and u.

Let (x,xx,x),x indicate that the pathway ends.

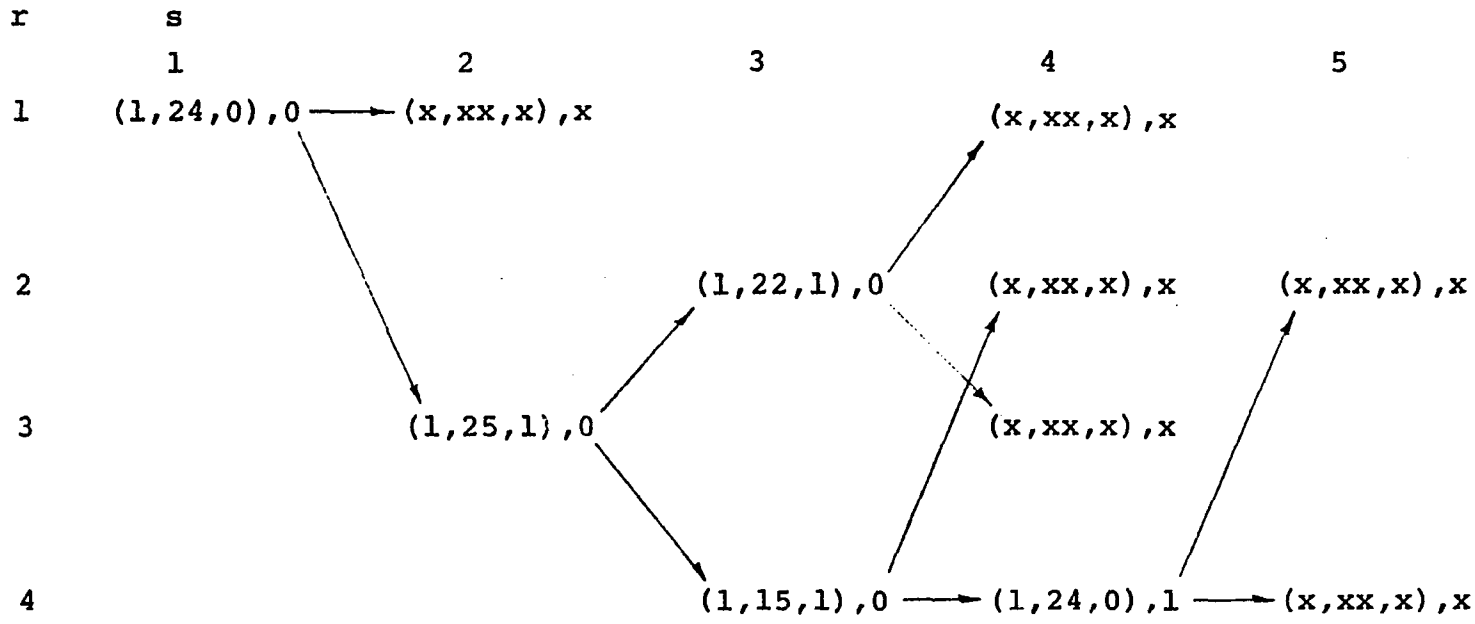


Figure 13. Decision pathways for Figure 12

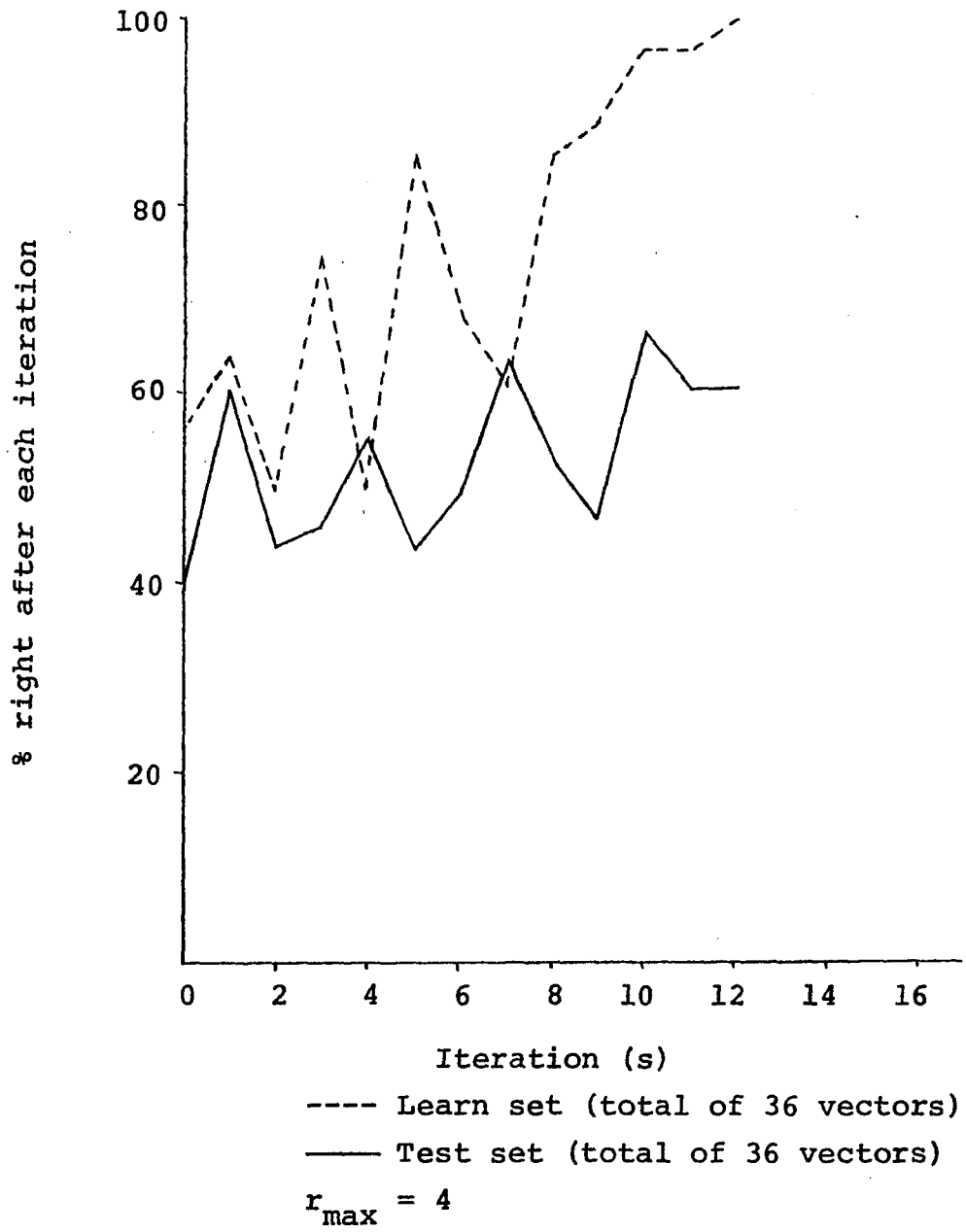


Figure 14. Example 2. EKG without QRS measurement

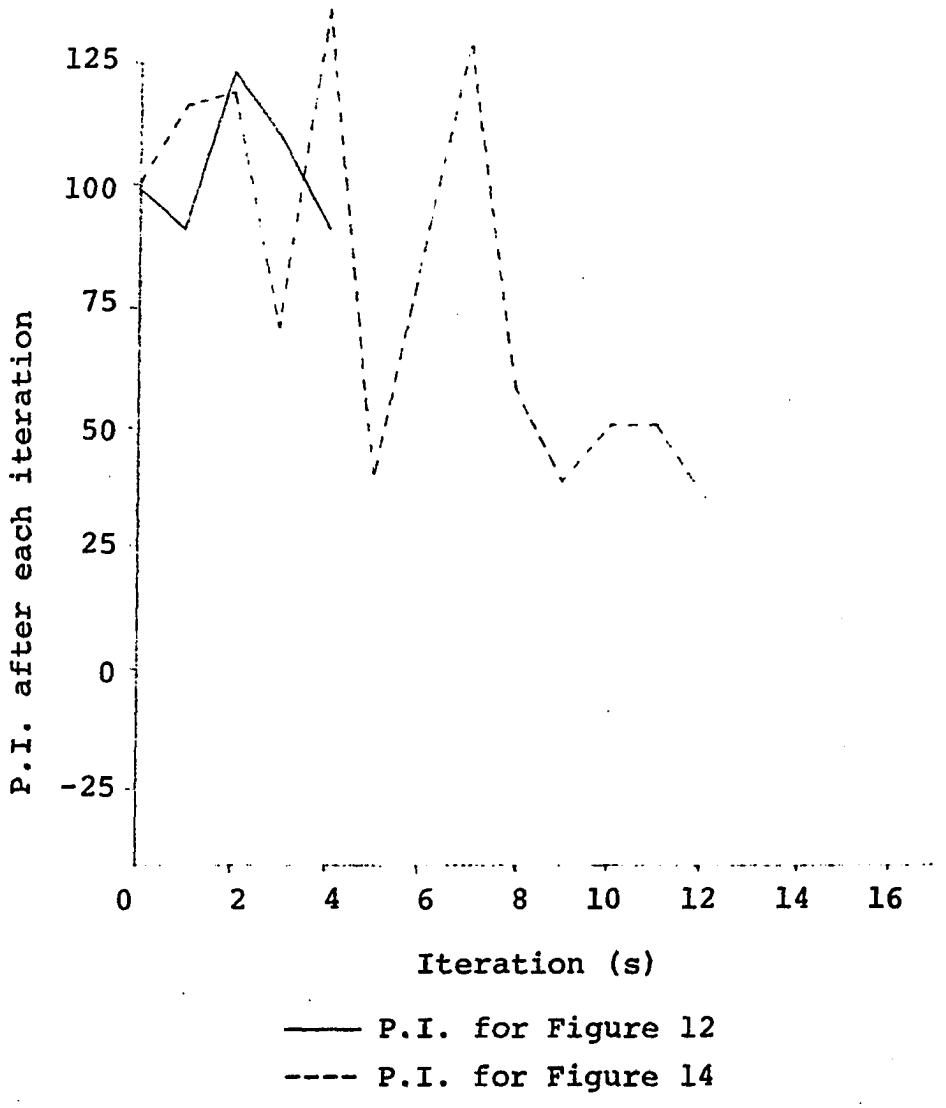


Figure 15. Performance indexes for Example 2

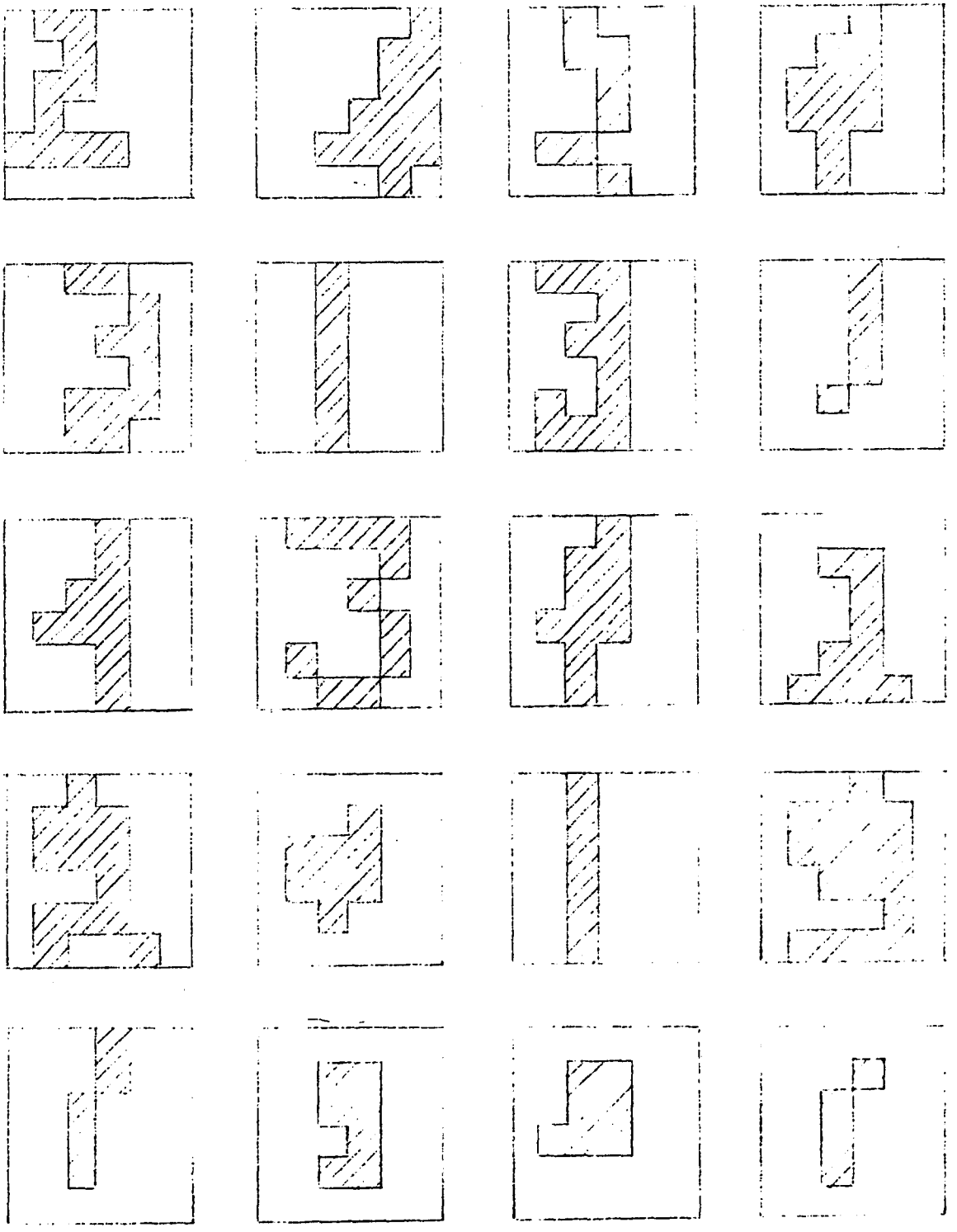


Figure 16. Samples of O vectors for Example 3

N vector

1					
7					
13					
19					
25					
31					

Feature extractor 1

Output 4 if $n_9 = n_{15} = n_5 = 0$.

Output 3 if $n_9 = n_{15} = n_{21} = 0$.

Output 2 if output is not 3 or 4 and if $n_{10} = n_{11} = n_{21} = 0$.

Output 1 if output is not 2 or 3 or 4 and if $n_{10} = n_{16} = n_{22} = 0$.

Output 2 if none of the previous statements hold, and if $n_2=0$, otherwise no classification.

Feature extractor 2

Output 4 if $n_3 = 0$.

Output 1 if $n_3 = n_{21} = 1$.

Output 2 if $n_3 = 1$, $n_{21} = n_{11} = 0$.

Output 3 if $n_3 = n_{11} = 0$, $n_{21} = 0$.

Feature extractor 3

Output 1 if $n_{20} = n_{26} = n_{32} = 0$.

Output 4 if output is not 1 and if $n_{21} = n_{20} = 1$.

Output 2 if output is not 1 or 4 and if $n_{27} = 1$, $n_{33} = 0$.

Output 3 otherwise.

Figure 17. Format for Example 3

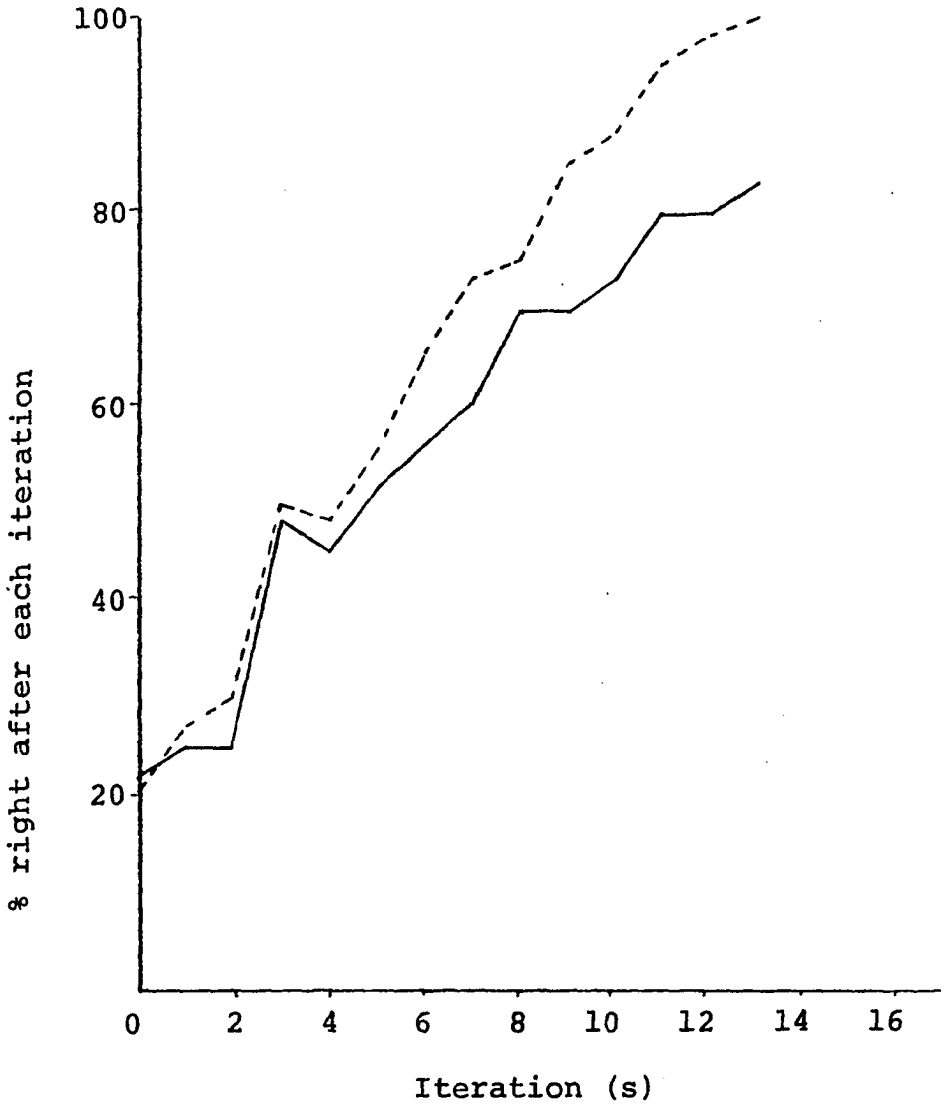


Figure 18. Example 3. Four numbers

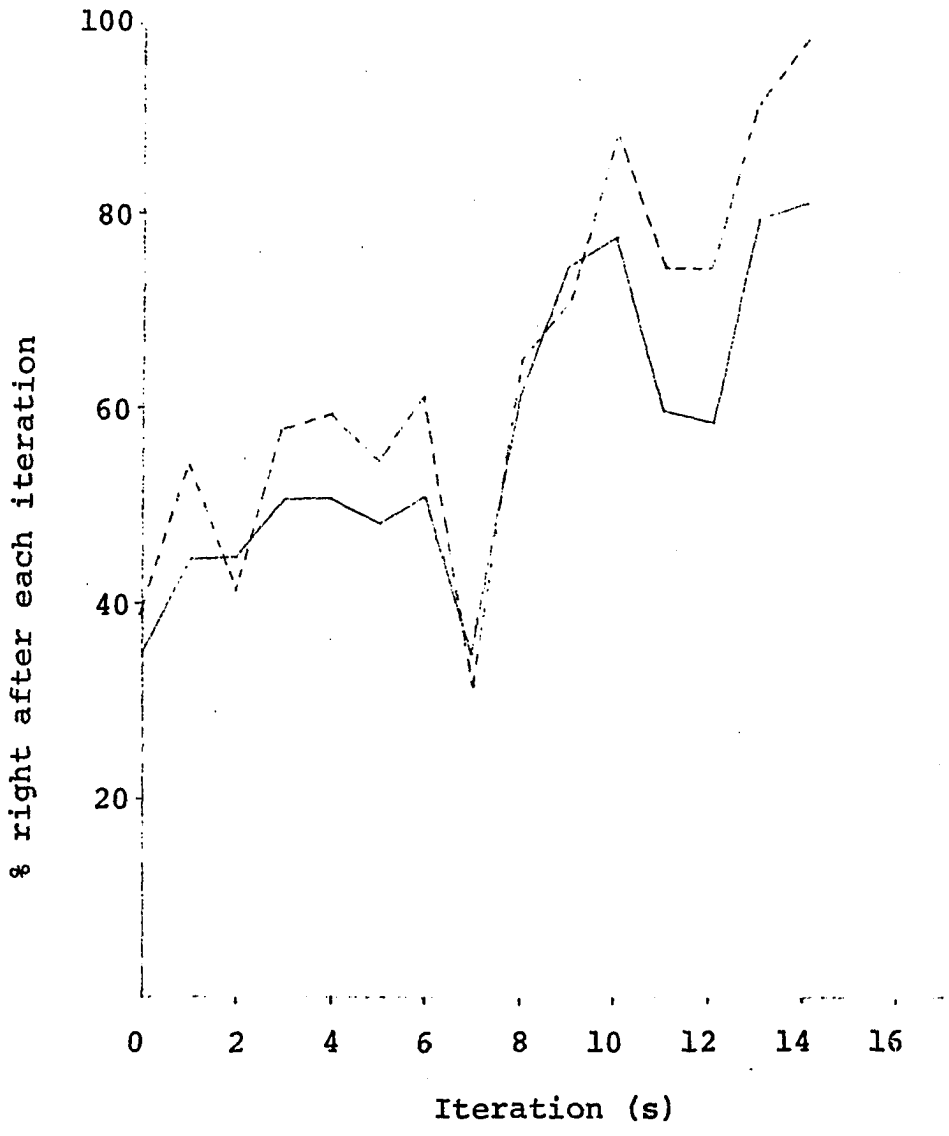


Figure 19. Example 3. Four numbers

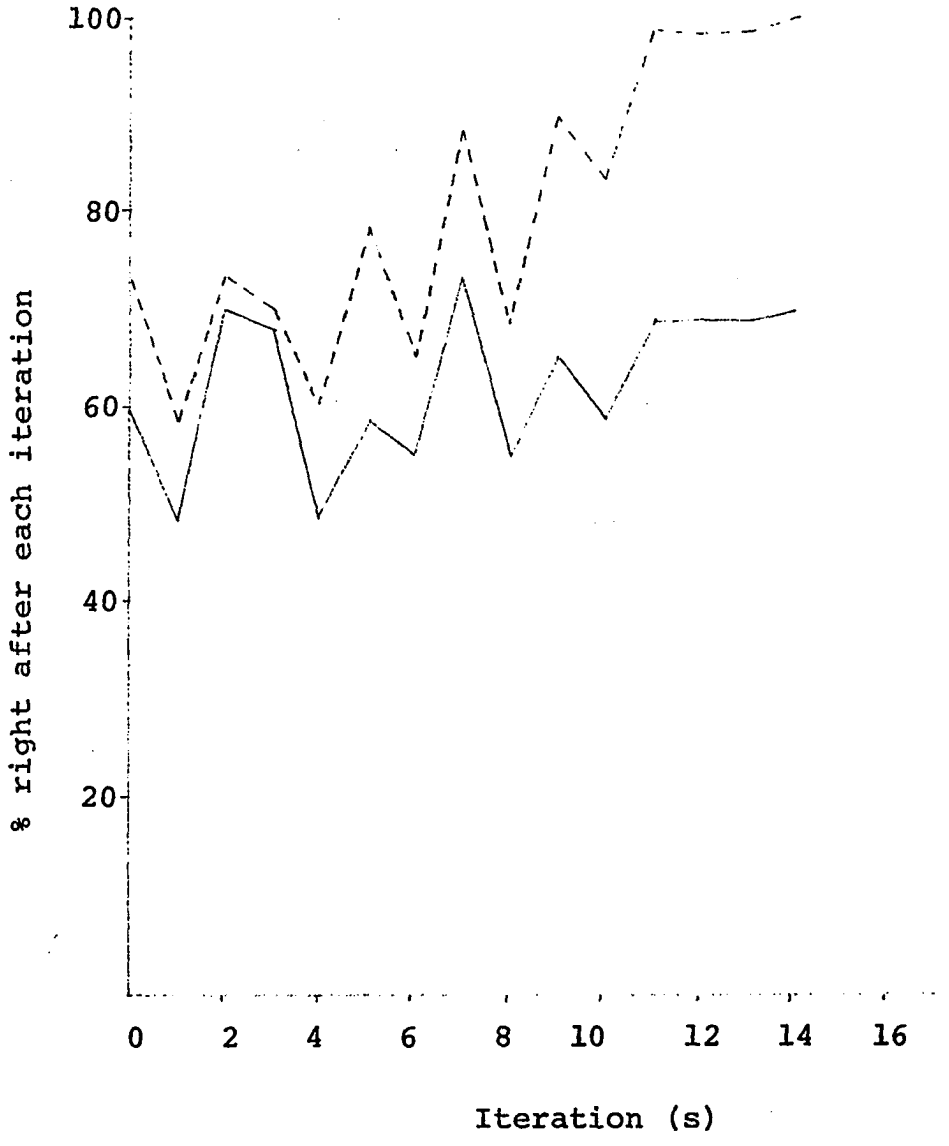


Figure 20. Example 3. Four numbers

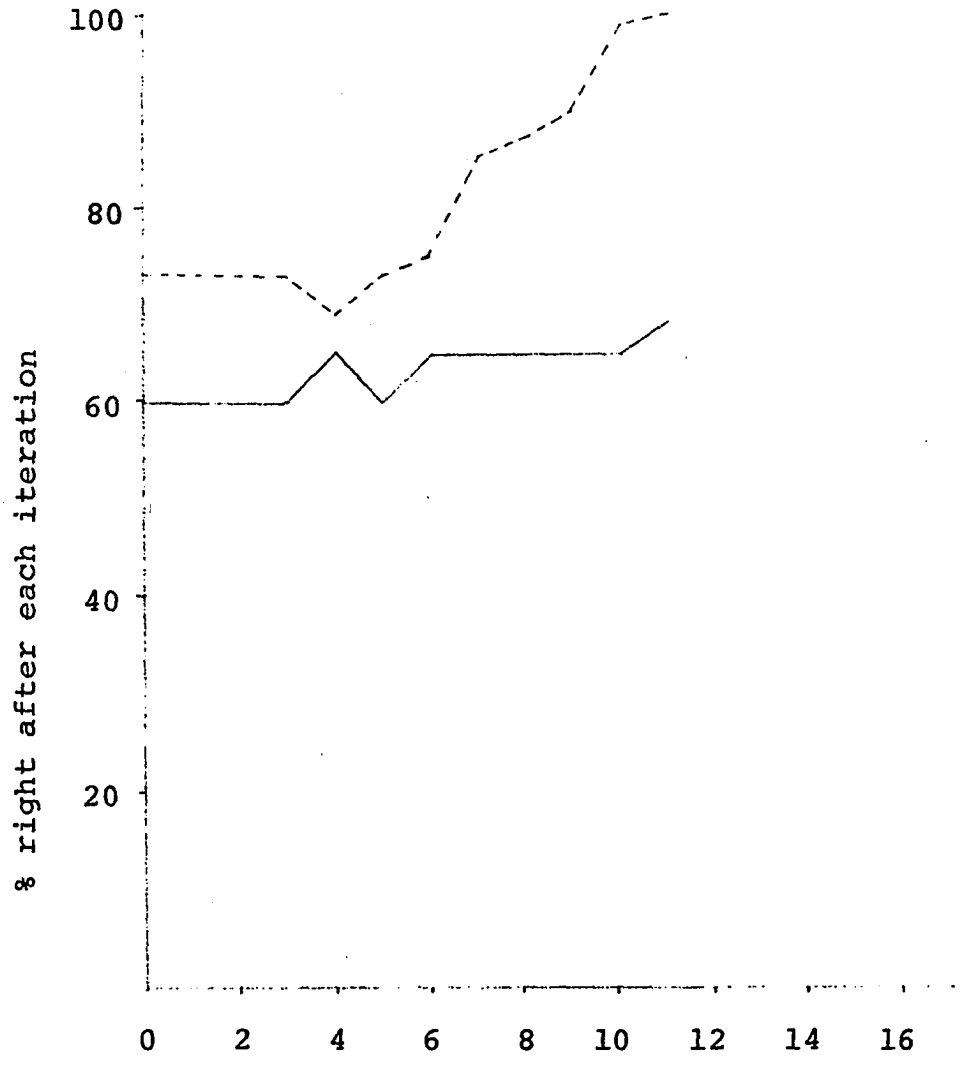


Figure 21. Example 3. Four numbers

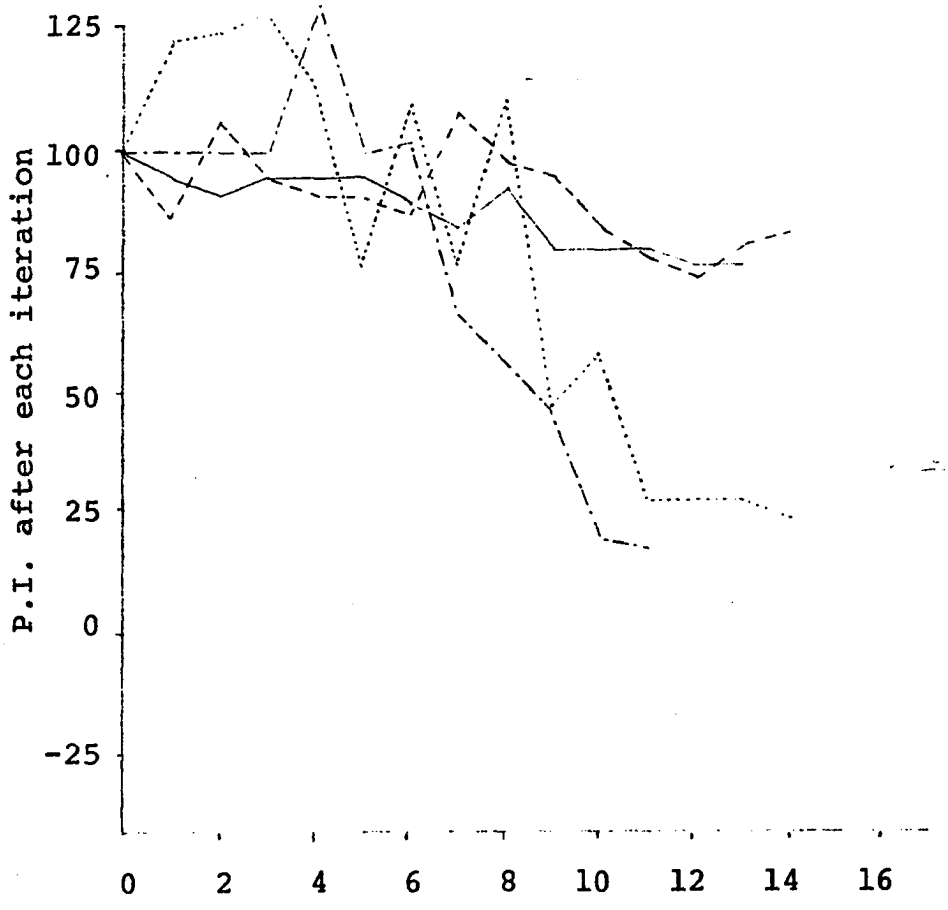


Figure 22. Performance indexes for Example 3

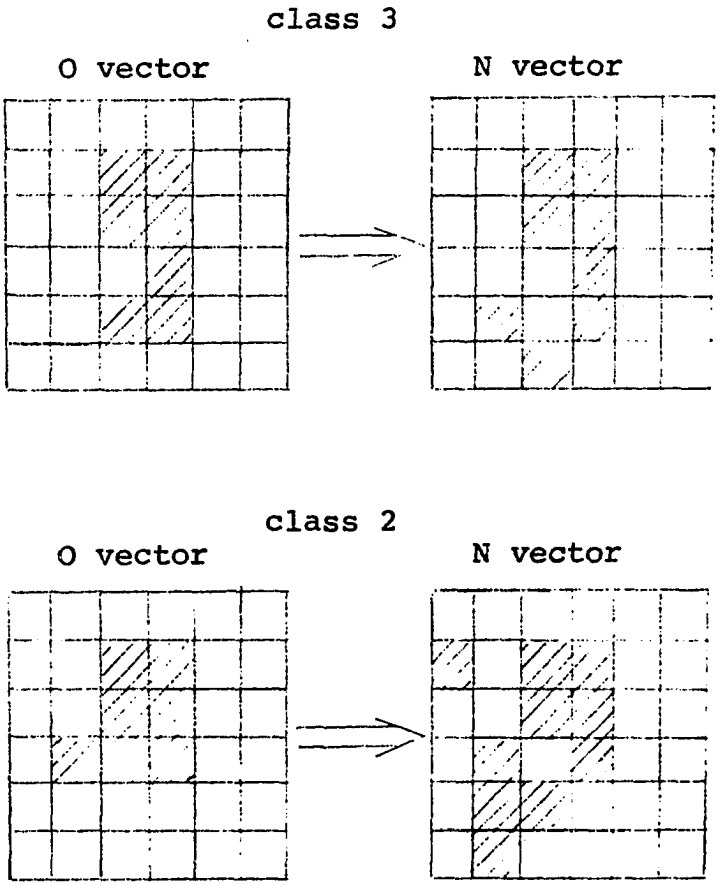
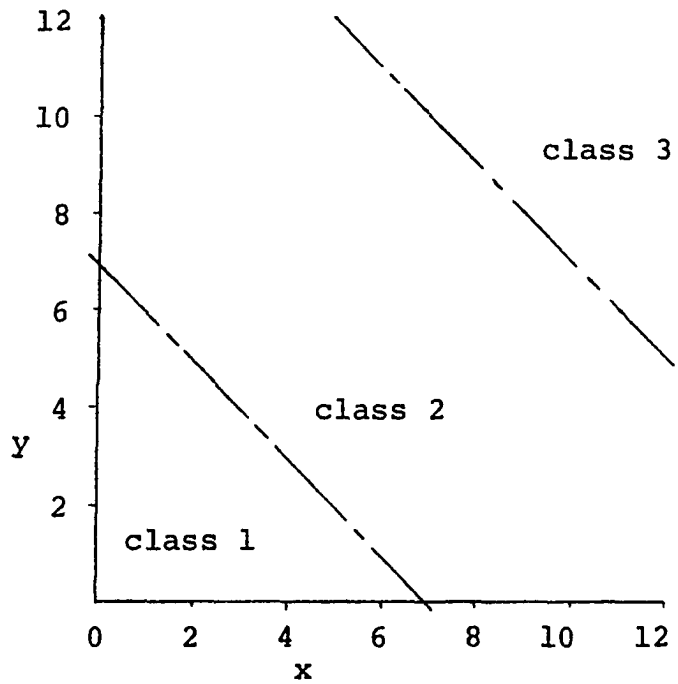


Figure 23. Transformations from Figure 20



Feature extractor 1

class 1 if $x \leq y-1$

class 3 if $x > y+1$

class 2 if $y-1 < x \leq y-1$

Feature extractor 2

class 1 if $x \leq 3$

class 3 if $x > 9$

class 2 if $3 < x \leq 9$

Feature extractor 3

class 1 if $2x \leq 10-y$

class 3 if $2x > 26-y$

class 2 if $10-y < 2x \leq 26-y$

Figure 24. Format for Example 4

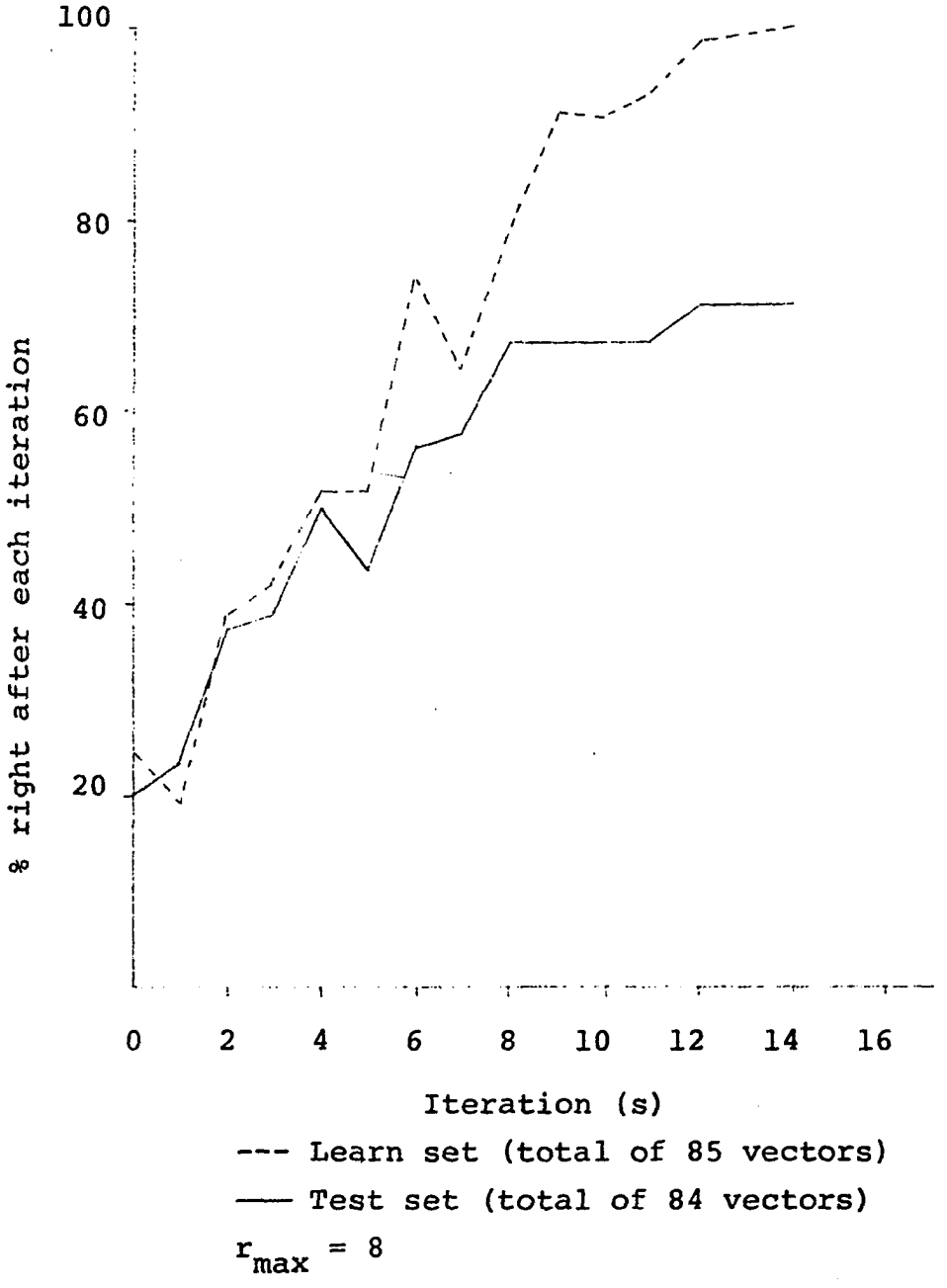


Figure 25. Example 4. Linear boundaries

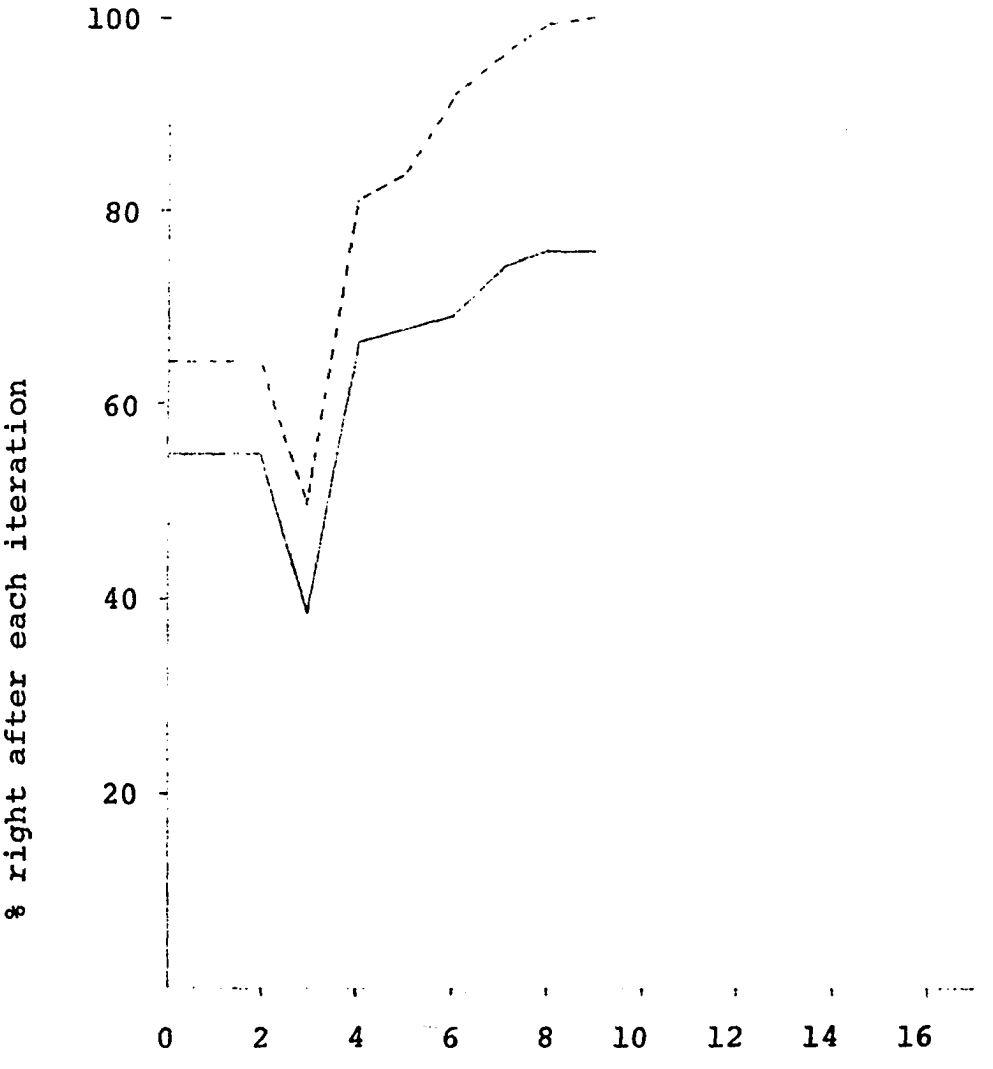


Figure 26. Example 4. Linear boundaries

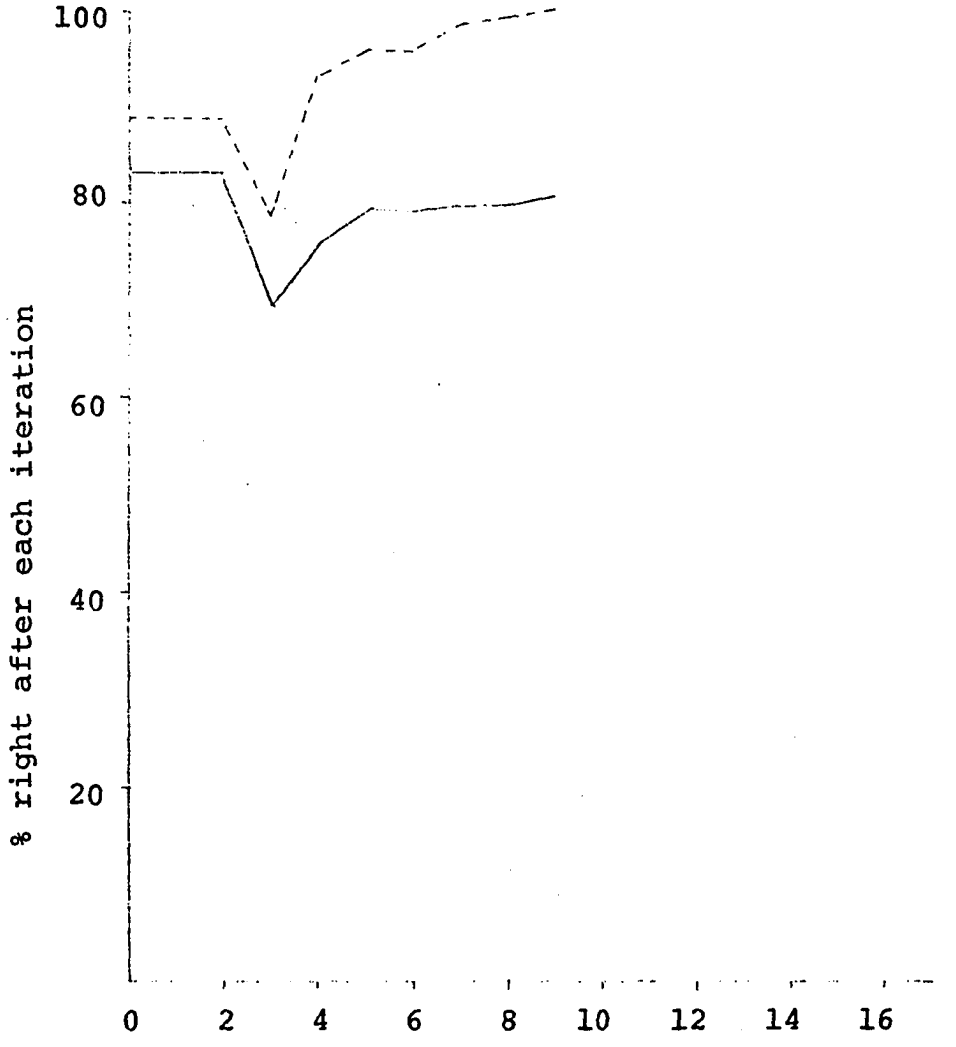


Figure 27. Example 4. Linear boundaries

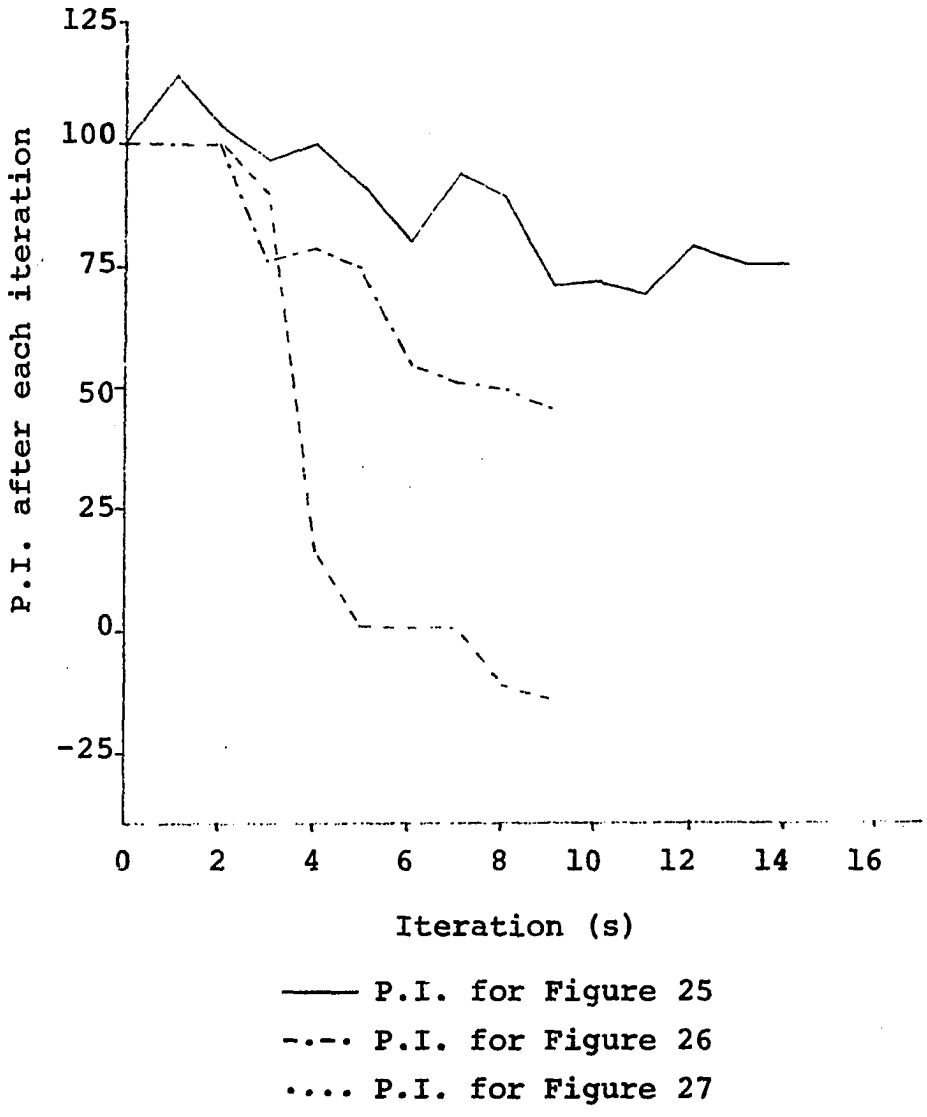
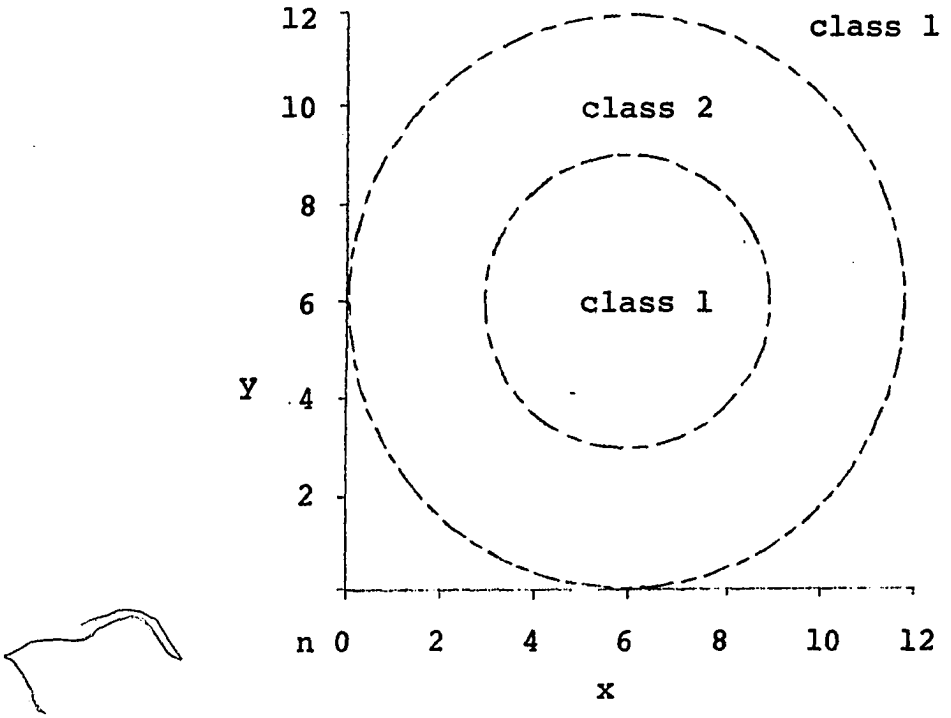


Figure 28. Performance indexes for Example 4



Feature extractor 1

class 1 if $x \leq 6$.

class 2 if $x > 6$.

Feature extractor 2

class 1 if $(x-6)^2 + (y-6)^2 > 36$.

class 2 if $(x-6)^2 + (y-6)^2 \leq 36$.

note: All errors are due to inner disc of class 1.

Figure 29. Format for Example 5

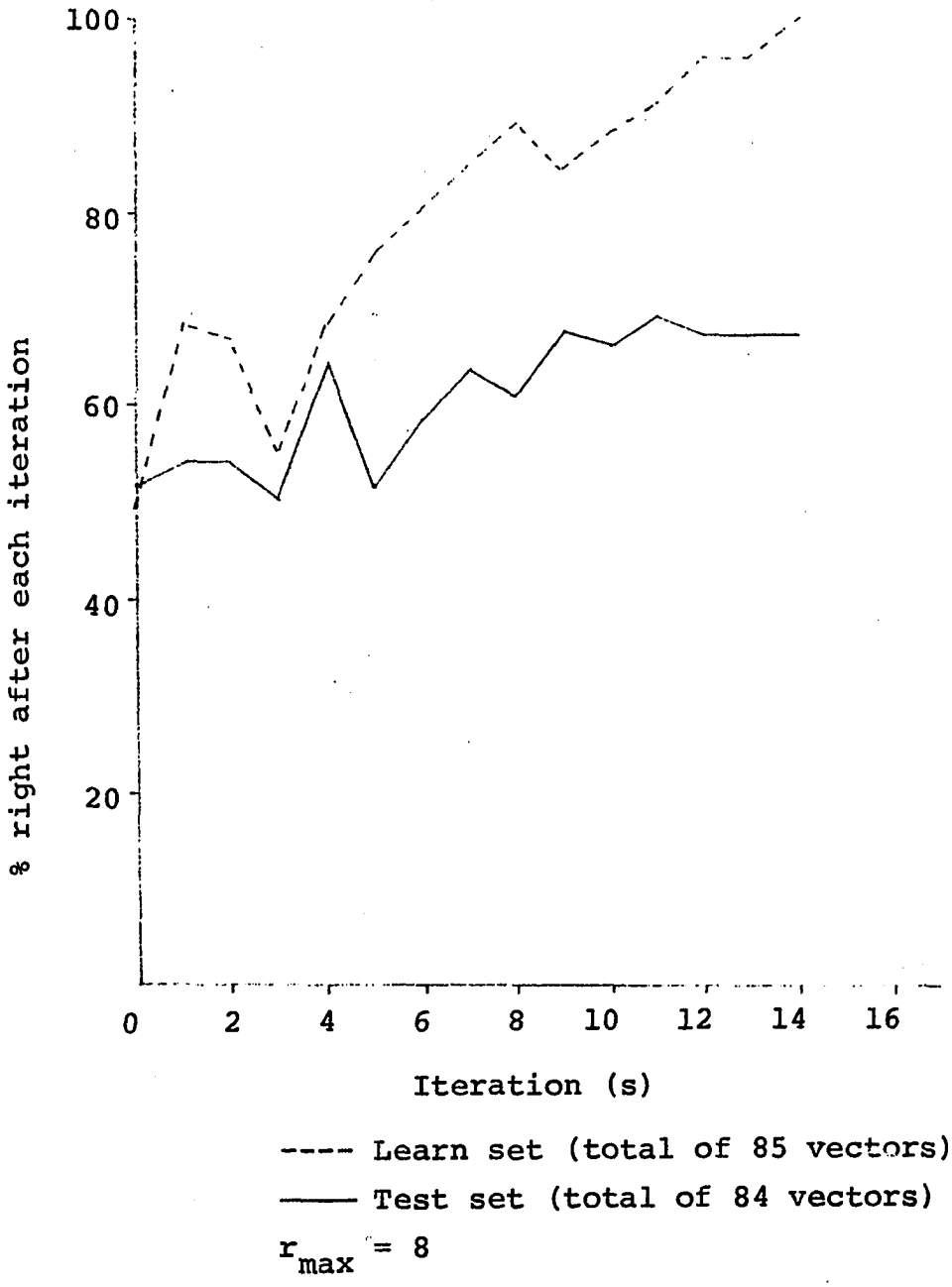


Figure 30. Example 5. Circular boundaries

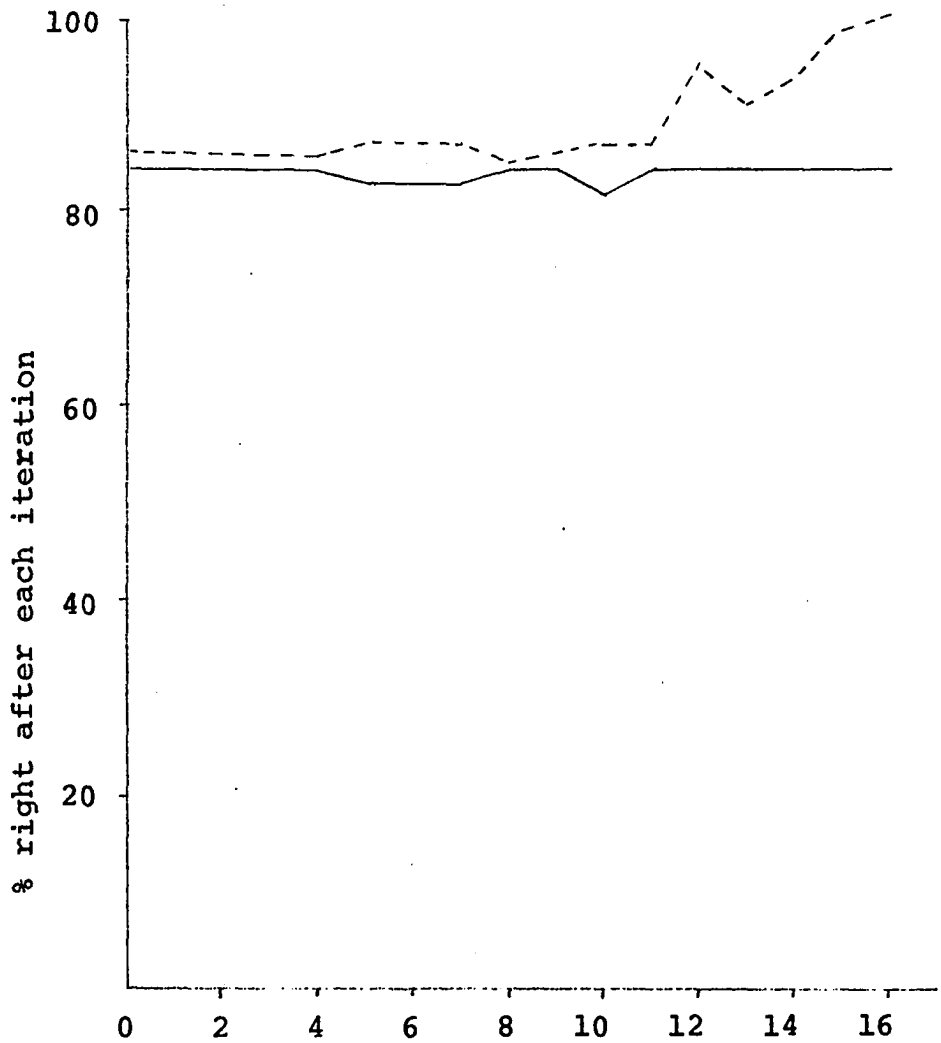


Figure 31. Example 5. Circular boundaries

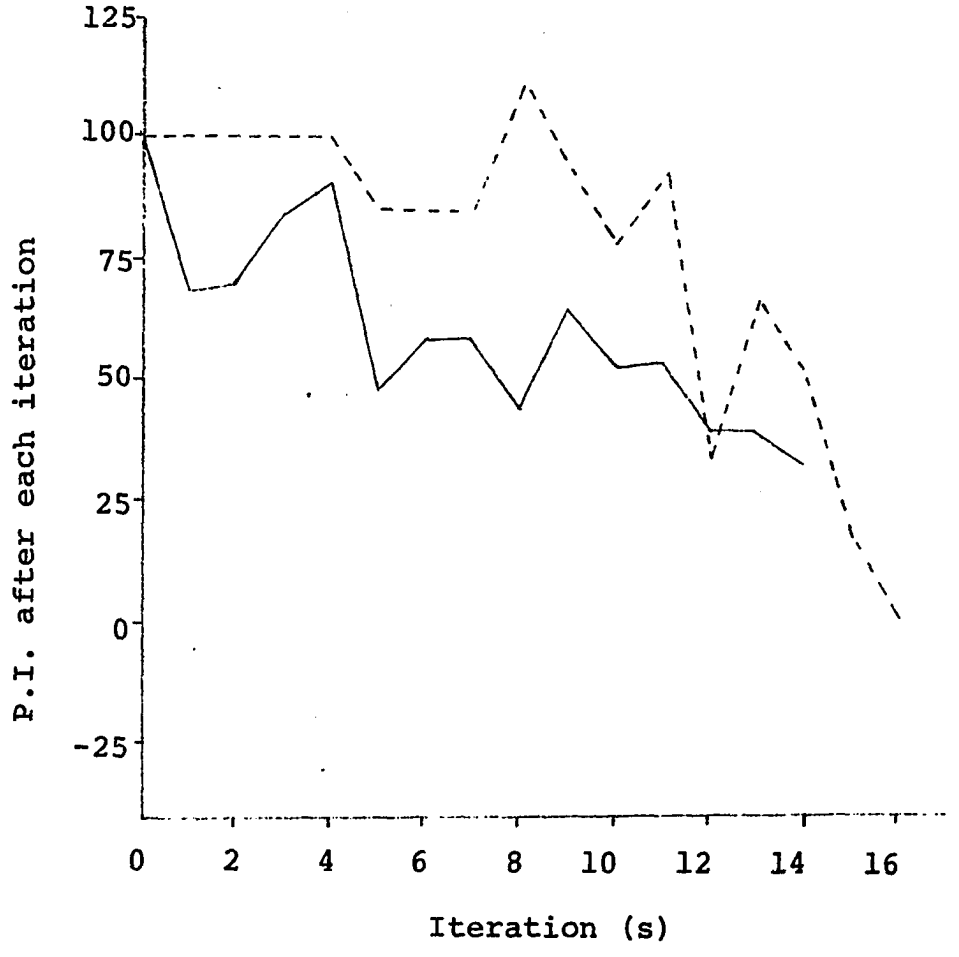


Figure 32. Performance indexes for Example 5

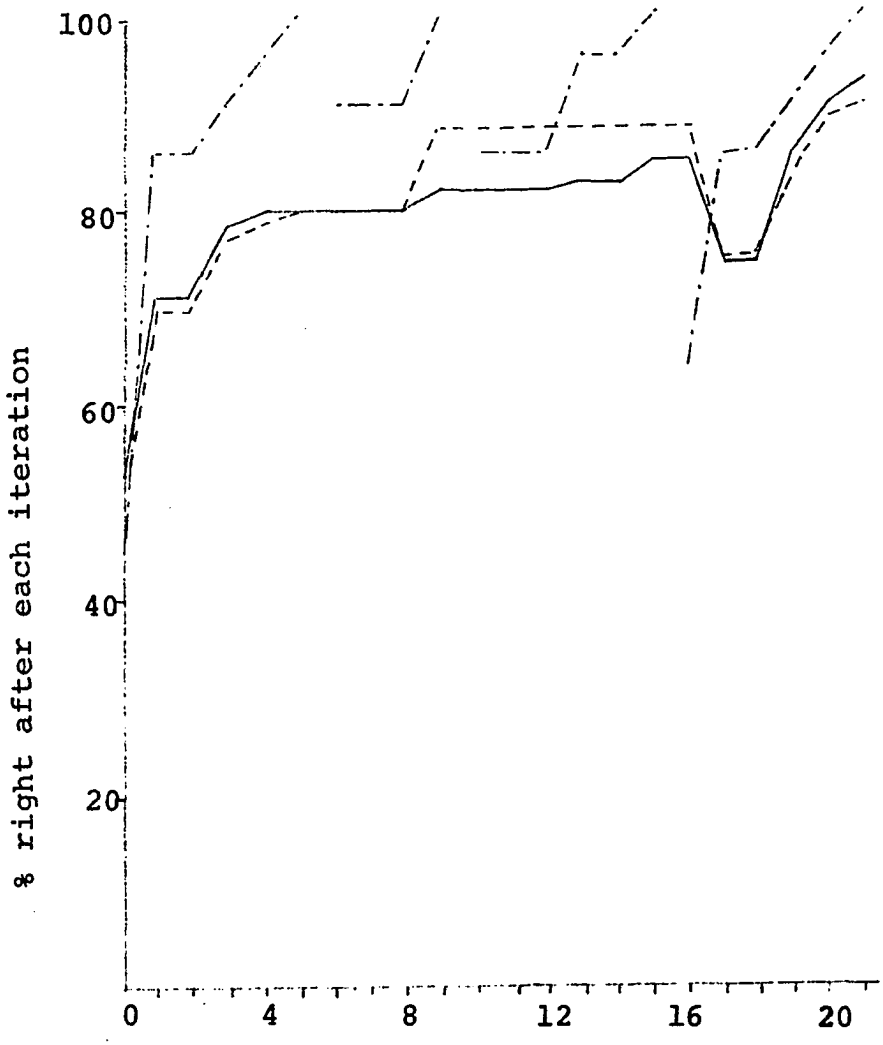


Figure 33. Example 6. Learning by subsets

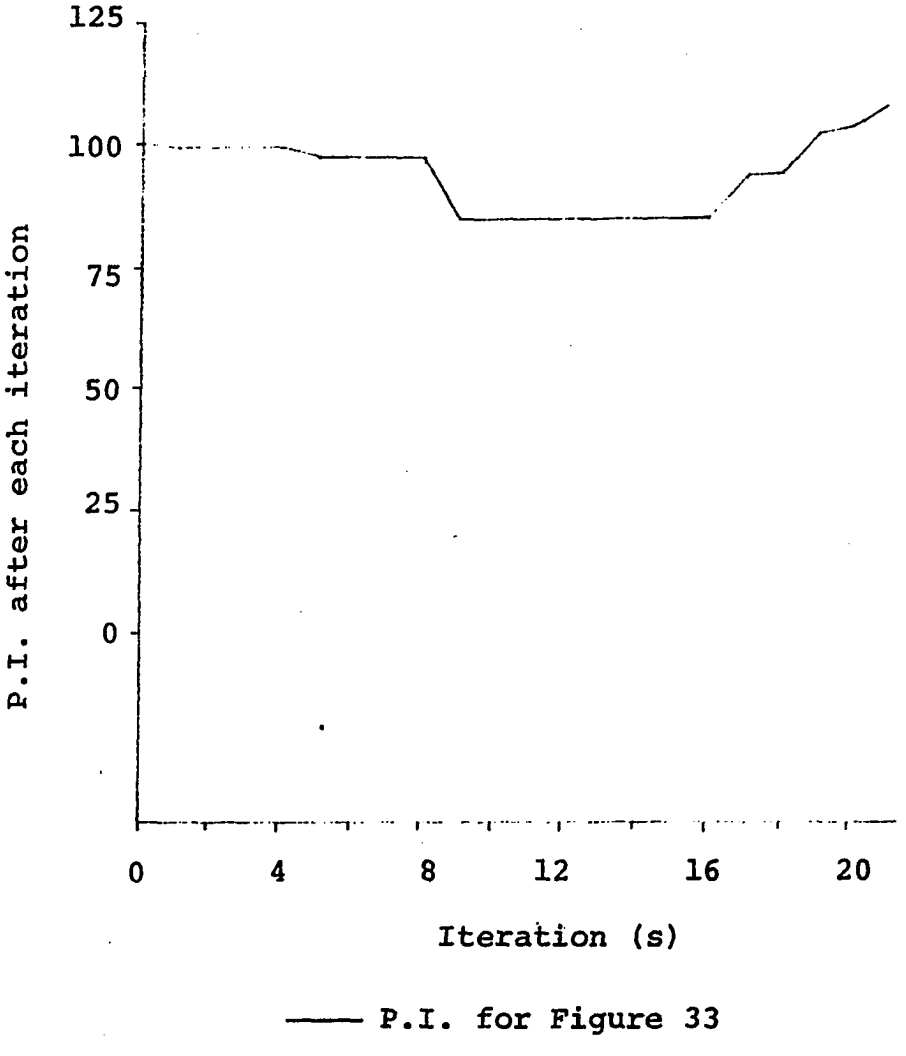


Figure 34. Performance index for Example 6

VIII. BIBLIOGRAPHY

1. Brockman, William H. A stimulus conditioning learning model and its application to pattern recognition. Unpublished Ph.D. thesis. Lafayette, Indiana, Library, Purdue University. 1966.

2. Chow, C. K. A recognition method using neighbor dependence. I.R.E. Transactions on Electronic Computers E.C-11: 683-690. 1962.

3. Chow, C. K. and Liu, C. N. An approach to structure adaptation in pattern recognition. I.E.E.E. Transactions on Systems Science and Cybernetics SSC-2: 73-80. 1966.

4. Cover, T. M. and Hart, P. E. Nearest neighbor pattern classification. I.E.E.E. Transactions on Information Theory IT-13: 21-27. 1967.

5. Graenias, E. C., Meagher, P. F., Norman, R. J. and Essinger, P. The recognition of handwritten numerals by contour analysis. I.B.M. Journal of Research and Development 7: 14-21. 1963.

6. Greenberg, H. J. and Konheim, A. G. Linear and nonlinear methods in pattern classification. I.B.M. Journal of Research and Development 8: 299-307. 1964.

7. Guyton, Arthur C. Textbook of medical physiology. Philadelphia, Penn., W. B. Saunders Co. 1966.

8. Harmon, Willis W. Principles of the statistical theory of communication. New York, N.Y., McGraw-Hill Book Co. 1963.

9. Horwitz, L. P. and Shelton, G. L., Fr. Pattern recognition using autocorrelation. I.R.E. Proceedings 49: 175-185. 1961.

10. Hu, Ming-Kuei. Visual pattern recognition by moment invariants. I.R.E. Transactions on Information Theory IT-8: 179-187. 1962.

11. Lewis, P. M. The characteristic selection problem in recognition systems. I.R.E. Transactions on Information Theory IT-8: 171-178. 1962.

12. Sebestyen, George S. Decision-making processes in pattern recognition. New York, N.Y., McGraw-Hill Book Co. 1963.

13. Sherman, H. A quasi-topographical method for the recognition of line patterns. International Conference on Information Processing Proceedings 1959: 232-238. June 1959.

14. Sprick, W. and Ganzhorn, K. An analogous method for pattern recognition by following the boundary. International Conference on Information Processing Proceedings 1959: 238-234. June 1959.

15. Tenery, George. A pattern recognition function of Integral geometry. I.E.E.E. Transactions on Military Electronics MIL-7: 196-199. 1963.

16. Tou, Julius T. and Heydorn, Richard P. Some approaches to optimum feature extraction. In Julius T. Tou, ed. Computer and Information Sciences-II. Pp. 57-89. New York, N.Y., Academic Press. 1967.

17. Turner, L. F. A system for the automatic recognition of moving patterns. I.E.E.E. Transactions on Information Theory IT-13: 21-27. 1967.

18. Unger, S. H. Pattern detection and recognition. I.R.E. Proceedings 47: 1737-1752. 1959.

IX. ACKNOWLEDGEMENT

I would like to thank Dr. William H. Brockman for his excellent suggestions and guidance given during the preparation of this dissertation. His insight into the pattern recognition problem and his critiques have been an immense help.