Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

A PATTERN RECOGNITION PROGRAM WHICH USES A GEOMETRY-PRESERVING REPRESENTATION OF FEATURES

by

Albert L. Zobrist

Technical Report # 85

March 1970

ABSTRACT

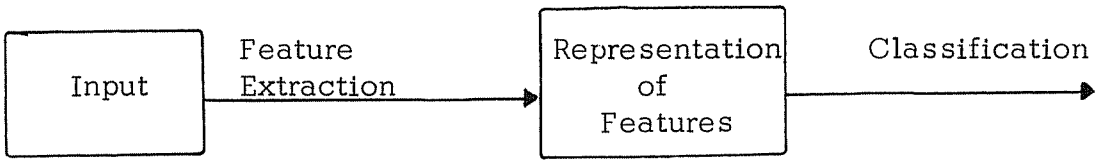
This report describes a two layered pattern recognition program. The first layer scans an input for features and produces a coded representation. The second layer looks for combinations of code which signify relations between features in the input. It is argued that the nature of the representation produced by the first layer determined the amount and quality of subsequent processing. Both layers create and modify their operators. Good results are obtained using the Highleyman hand printed data. The program is discussed as a model of an aspect of human perception.

INTRODUCTION

Feature extraction is a popular method for general pattern recognition primarily because it is relatively easy to implement and because it has worked as well as any method. A large body of computer algorithms and philosophies dealing with feature extraction has been surveyed recently by Levine[1]. Another reason for the appeal of this method is that features appear to be the basic elements of human perception.[2] This report suggests a more effective organization of features for such programs and describes a working program which uses this suggestion. The program is discussed as a model of perception with the hope of giving insight into human pattern recognition capabilities.

FEATURE EXTRACTION PROGRAMS

For purposes of discussion, we shall consider pattern classification to be a two stage process:



Feature extraction attempts to characterize the input by a set of properties which allow efficient classification. The following paragraphs summarize this aspect of two computer programs, paying special attention to the manner in which features are represented or stored.

The Uhr-Vossler program[3] creates its own characterizers according to a fixed set of rules. The characterizers are 5 x 5 matrices of zeros, ones, and don't-cares which can be scanned over an input for possible occurrences. In addition, there are compound characterizers which are boolean combinations of several 5 x 5 matrices.

For each application of a simple characterizer the following information is stored: how many times it matched the input, the average i,j coordinate of these matches, and their mean square distance from the center. For compound characterizers, only the number of matches is stored. This information is in the form of a list containing the occurrence of features with some positional information.

The Munson program [4] has a preprogrammed set of characterizers which detect the presence and location of edges and the presence, size,

location, and orientation of enclosures, concavities, and stroke tips. The edge information is stored in binary vectors with the bits indicating the general location (nine regions) and orientation (six 30^{o} intervals). The other features are measured by numerical quantities called descriptors. These descriptors are then transformed to bits of information in a binary vector. These vectors are the basis of further processing to classify the input.

Most programs which produce such lists or vectors proceed with a final step, more algorithmic in nature, to classify the input. Typically, n-tuples from the list or vector are used to give weights of implication or else a metric is placed on the vectors to find the distance of an unknown input to vectors obtained from prototypes. Nilsson[5] has written an introduction to this subject.

If we consider features to be the elements of pattern recognition, then we should consider methods which make use of the structure and organization of features in a scene. Thus we should ask whether present classification algorithms can utilize physical relations in a scene which may be important to pattern recognition. Such research will not just involve these algorithms, however, but will include study of the representations upon which they operate. Clearly, the types of processing which can practically be done depends upon the structure of the representation on which they operate. For our pattern classifiers, the information avail-

able to the final decision step must either be stored in the representation or calculated therefrom. This is an argument for a more complex representation of features (even if no more information is added to the representation) which allows more information to be calculated.

A PROGRAM WHICH USES A GEOMETRY-PRESERVING REPRESENTATION OF FEATURES

The program is written in FORTRAN-V for the Univac 1108 computer. It is a pattern classifier, so it must be told the number of characters in the alphabet and their names. All experiments reported here were performed on the Highleyman data[6] which consists of 50 alphabets of hand printed characters, each alphabet having 10 numerals and 26 upper case letters. Each character is quantized and encoded as a 12 x 12 binary array. Its operation can be outlined as follows:

Step I. Uhr-Vossler characterizers are applied to the input. Whenever they match, a mark is placed in the corresponding point of an array reserved for that characterizer.

Step II. N-tuples are scanned over the arrays produced by Step I. Each n-tuple requires its marks to be found in their arrays at fixed relative position. The n-tuples imply the names of the alphabet with certain weights.

Step III. The weights are summed. The name with maximum weight is chosen as a response.

Step IV. The program is given the correct response as feedback. The values and the weights of implication of the n-tuples are adjusted up or down, and the values of the characterizers are also adjusted.

Step V. If the feedback is different from the response, then new characterizers and n-tuples are extracted from the input. Characterizers and n-tuples with low value may be discarded.

These steps will be examined in detail. We shall consistently use the terms characterizer and n-tuple as above, even though the n-tuples could be called compound characterizers. The program can start with null memory and create all n-tuples and characterizers or it can be given a pre-programmed set of characterizers and/or n-tuples. Note that weight refers to weight of implication of an n-tuple, and value refers to the worth accumulated by a characterizer or n-tuple.

Steps I and II are illustrated by Figure 1. The Uhr-Vossler characterizers are 5 x 5 arrays of zeros, ones, and blanks. Each characterizer is scanned over the input, and when the zeros and ones all match then a bit is placed. The placement is at the row and column corresponding to the center of the 5 x 5 array during a hit. For the results shown in the next section, between 64 and 84 characterizers were used. The 64 to 84 arrays produced by their application will be called the internal representation of the input.

An n-tuple consists of n references to the internal representation together with a specification of the geometric arrangement of the elements of the n-tuple. The 2-tuple shown in Figure 1 can be expressed as follows:

(0,0) characterizer 1

(3,-1) characterizer 2

Each n-tuple is scanned over the proper arrays to find a match, but no rotations or reflections are used. The geometric relations are allowed to

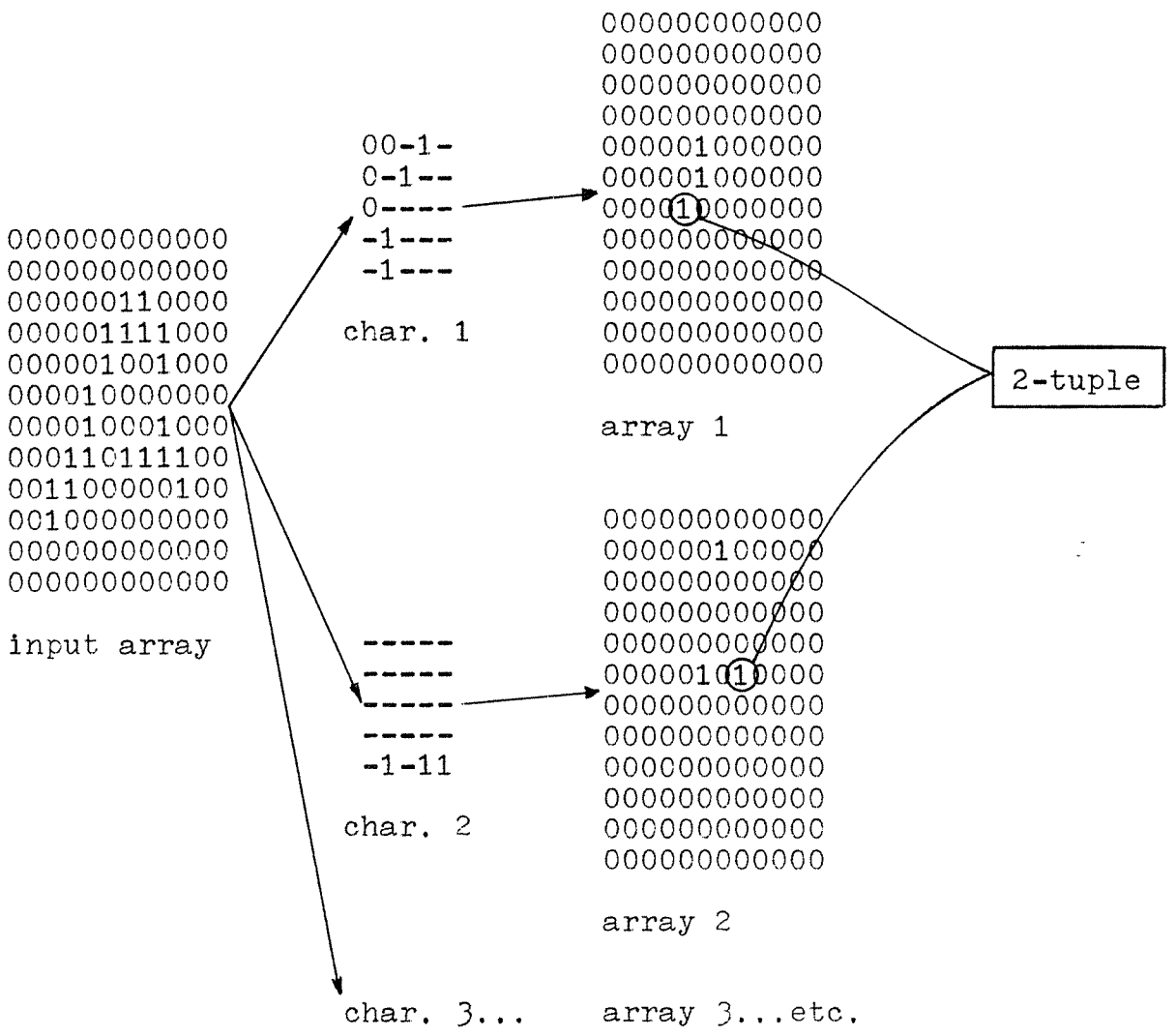


FIGURE 1. Illustration of the matching of characterizers and n-tuples

wobble a distance of 1 . For example, the bit placed by characterizer 2 could be at relative location (4,-2) in Figure 1.

Each n-tuple has an associated weight vector contining one weight of implication for each letter of the alphabet. If all parts of an n-tuple match, then its weight vector is added to a total sum vector. An n-tuple is allowed to add its weight only once per input. The highest value in the total sum vector determines the response of the program. The program is now given the correct name of the input as <u>feedback</u>. Steps IV and V are optional for a given run, so assume that they are requested.

If the feedback is different from the response, then step IV is performed as follows. Let T_k be an n-tuple which participated in the decision and suppose that it implied the response and feedback with weights W_{kr}, W_{kf} respectively. T_k may have made a positive or negative contribution, denoted $C_k = W_{kf} - W_{kr}$. If we let V_k be the value of T_k, then the following calculations are made:

$$X = \begin{cases} 10 & C_k \geq 0 \\ |C_k/10| & C_k < 0 \end{cases}$$

$$Y = \begin{cases} C_k & C_k \geq 0 \\ C_k/2 & C_k < 0 \end{cases}$$

$$V'_k = V_k + Y$$

$$W'_{kf} = W_{kf} + X$$

$$W'_{kr} = W_{kr} - X$$

Where the prime indicates that these new values are stored in memory to replace the old values. Thus, T_k will gain in value if it was "correct" in this case or will lose value if not. Note that any loss in value is divided in two. This is an ad hoc device for rewarding n-tuples which apply frequently. As each n-tuple is adjusted, the quantity $V'_k - V_k$ is added to the value of each characterizer to which that n-tuple refers. The weights are adjusted up 10 or down 10% so they will stabilize at a value which reflects the percentage of correct predictions. That is, if an n-tuple discriminates A from R correctly 75% of the time then the difference in weights for A and R will tend towards 300 .

If the response is correct, then the program still adjusts weights and values to improve discrimination. It pretends for the moment that the second highest weight corresponds to the machine's response, then it performs the calculations described in the previous paragraph. Thus the second highest response is inhibited and the correct response is reinforced.

Step V is performed only if the machine gives an incorrect response. One characterizer and three n-tuples are created according to the following procedure:

1) A 5 x 5 matrix is extracted from a random position in the input matrix. It is rejected if less than four "one" cells occur.

2) All "zero" cells adjacent to "one" cells are replaced by blanks. Each of the remaining nonblank cells are replaced by blanks with probability

$\frac{1}{2}$. The characterizer is rejected if less than two "ones" remain.

3) The value of the characterizer is set to the minimum value over all characterizers plus 100, or to 1000 if no characterizers exist.

4) One n-tuple is created by choosing at random n different characterizers which applied to the input and then choosing one bit at random from each of the arrays produced by those characterizers. The relative position of these bits determine the geometric structure of the n-tuple.

5) The n-tuple is given a weight vector of zeros except for the weight which corresponds to the true name of the input. That weight is set to 10x (size of alphabet).

6) The value of the n-tuple is set to $(V_a - V_m)/2$ where V_a is the average value and V_m is the minimum value over all n-tuples, or to 1000 if that value is less than 1000 .

7) Two more n-tuples are created as in steps 4-6 except that one of the references of each is assigned to the new characterizer just created, and to the geometric position from which it was extracted.

8) If there are more characterizers than the maximum allowed, then the characterizer of lowest value is discarded. All n-tuples which refer to it are discarded as well.

9) The most active characterizer is chosen by the formula h-s+32*m, where h is the number of hits scored by n-tuples which reference the

characterizer, s is its age in terms of the number of inputs, and m is the number of n-tuples which refer to it. This characterizer is penalized for its activity by discarding the n-tuple of lowest value which refers to it.

10) If there are still more n-tuples than the maximum allowed, then the n-tuple of lowest value is discarded.

Although the description given in this section appears complicated, the program is really quite simple, requiring about 600 FORTRAN statements. It is reasonably fast, requiring .3 seconds per input for the Highleyman data. The principal drawback is the large amount of core storage required for the arrays used by the characterizers. This problem might be eased by placing some or all of the characteristics in the same array, using different symbols for different features. A program written by Forsen[7] uses one array to represent lines and edges extracted from the input by symbols which indicate the orientation. For our purposes, a certain point in the input may belong to several features, thus the marks will cancel one another. If we desire a rich computer representation of features in a reasonable amount of storage, then more thought must be given to this matter.

RESULTS

The Highleyman data has been used in a variety of pattern recognition experiments[8]. Some have used the first 40 alphabets as training data to determine parameters and the last ten alphabets as testing data, with no parameter adjustment during the testing phase. This arrangement has been followed to produce the results shown in Figure 2. The specific conditions for the three runs marked ABC are as follows:

1) In all three runs, the program used only 2-tuples, all of which were created by the program, allowing a maximum of 400.

2) Runs A and B used 64 first level characterizers which were designed by the author.

3) For run C , the program created all of its own characterizers, allowing a maximum of 84. Note that the results are superior to run B , which used predesigned operators.

4) A 34 letter alphabet omits the numerals 0 and 1 . A 10 letter alphabet consists of the numerals 0 through 9 .

When a data set as small as Highleyman's is used, the scores must be interpreted carefully. For example, one might obtain 100% success on the training data by the use of 1400 characterizers, each of which correctly identifies one training sample. Thus the independent test score is a better measure of success. By this measure, Munson has achieved the best results to date. If we consider these programs to be learning machines, then another

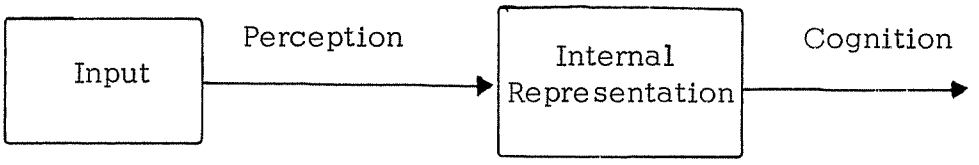
important measure is the generalization, the difference between the training score and the test score. In both of these respects the present program performs very well.

	size of alphabet	# training passes	training score	test score
Chow	36	1	93.3	58.3
Munson	36	18	-	68.3
A.	34	4	49.4	43.8
Munson	10	18	-	88.0
B.	10	12	76.8	63.0
C.	10	10	83.3	75.0
Human perf.	34	-	-	88.5

FIGURE 2. Recognition rates for experiments with Highleyman's data.

THE PROGRAM AS A MODEL OF PERCEPTION

In an early paper[9], Greene pointed out that computers of that day did not contain the meaning of the propositions they handle. "Every proposition is reduced to a hole or an electric charge or a magnetic field at some point of space. Thus the propositions may be sorted and combined but their meaning resides in the mind of a person who looks at a list which says that a hole in this place means that proposition." But surely the human mind has its equivalent of bits and holes. How can we say that a certain distribution of matter and energy in the brain has meaning? If we take "meaning" to mean the internal connections, then Greene is saying that a symbol may have more connections in the programmer's mind than in the computer program which manipulates it. I would like to argue that a perception carries meaning to our mind, and that the meaning resides in the representation caused by the perceptive processes. Furthur processing occurs, more cognitive in nature, to give us the end result of our perception. We shall call this a model of perception. It is rather vague and doesn't have many working parts, but it is a useful framework for the discussion which follows.

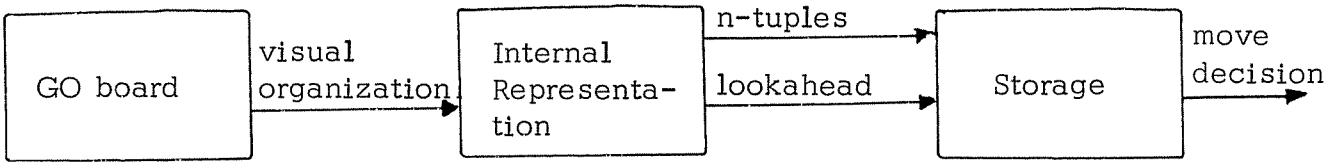


The richness of the representation determines the amount and quality of furthur processing which may occur. To quote from Greene again, "Most

of our perception is too rich to be described in what is called a discursive code, a one-dimensional sort of language that can be written on a typewriter or spoken or put on a magnetic tape." Thus it appears that a complex representation of perceptual features is necessary for the "mind's eye."

Feature production programs can all be thought of as realizations of this model. In particular, the program described in that last section is an attempt to improve pattern recognition performance by enriching the representation of features. More specific hints for the program were provided by the studies of Hubel and Weisel[10] in which nerve impulses were found in the occipital region of the brain of cats in response to a variety of stimuli which correspond to "features" such as spots and edges. Similar transformations were studied in the retina of frogs by Lettvin et. al.[11] In man it appears that many parallel sets of transducers operate on a visual input to produce the features which carry meaning to our mind. Their connections are interwoven so that the impulses which represent features maintain their geometric position. This arrangement has been modeled by mapping the results of the first level characterizers into arrays of the same dimension as the input array. Although the location of occurrences of features could be placed in a list, it would then be difficult to reference this information by location. This type of processing is prohibited by the Uhr-Vossler and Munson programs.

The authors GO program[12] uses a similar representation for features extracted from the input, which is an array of black and white stones. Again, we have perceptive processes followed by processes more cognitive in nature.



This program is the only successful GO playing program reported in the literature and is capable of winning against a weak human opponent. We should be very interested when two programs operating on similar principles achieve success in different tasks. Generality may be achieved by combining such programs in a non-trivial fashion. A suitable representation of extracted features might enable calculations which correspond to a wide variety of intelligent activities. It has been suggested that list structures or symbolic calculi are suitable for this purpose[13], but I am suggesting human perception as the prototype for an input to artificially intelligent devices.

To date, the performance of these programs is the only objective comparison of them, as models, to their biological prototypes. This serves the goal of efficiency of operation. However, there is the exciting prospect of new comparisons of behavior as the models become more detailed. For example, psychological phenomena which are thought to occur primarily in the early layers of the visual system, such as masking and stereopsis, might be obtained with models not radically different than those discussed here. The goal now is knowledge about the human mind. In my opinion, striving towards the second goal is the best means of achieving the first.

SUMMARY

The Uhr-Vossler program is notable for its creation and adjustment of its own characterizers. The program described in this report inherits these advantages, and is capable of creating a second level of characterizers (n-tuples) as well. The second level characterizers can assess geometric relations between the features detected in the input. This is made possible by a geometry-preserving representation of features at the first level. This program, as well as the Uhr-Vossler program, can be considered as models of human perception. Thus, the good results reported above are achieved by a series of programs which express theoretical advances as well.

ACKNOWLEDGEMENTS

I would like to thank Professor Leonard Uhr for his help and guidance. This work was conducted with the support of NIH grant MH12266 and NSF grant GP7069.

BIBLIOGRAPHY

1. M. LEVINE, Feature extraction: a survey, Proceedings of the IEEE, 57 (8), 1391-1407, Aug. 1969.

2. U. NEISSER, Cognitive Psychology, Appleton-Century-Crofts, New York (1967).

3. L. UHR and C. Vossler, A pattern recognition program that generates, evaluates, and adjusts its own operators, Proceedings of the WJCC (1961).

4. J. H. Munson, Experiments in the recognition of hand-printed text: part I - character recognition, Proceedings of the FJCC, 1125-1138 (1968).

5. N. J. NILSSON, Learning Machines, McGraw Hill, New York (1965).

6. W. HIGHLEYMAN, Data for character recognition studies, IEEE Transactions on Electronic Computers, EC-12, 135, Apr. 1963.

7. G. E. FORSEN, Processing visual data with an automaton eye, in Pictorial Pattern Recognition, Thompson, Wash. D. C. (1968).

8. J. H. MUNSON, R. O. DUDA, and P. E. Hart, Experiments with Highleyman's data, IEEE Transactions on Electric Computers, EC-14(4), 399-401, Apr. 1968.

9. P. H. GREENE, An approach to computers that perceive, learn, and reason, Proceedings WJCC, (1959).

10. D. H. HUBEL and T. N. WEISEL, Receptive fields of single neurons in the cat's striate cortex, Journal of Physiology, 148, 574-591 (1959).

11. J. Y. LETTVIN, H. R. MATURANA, W. S. MCCULLOCH, and W. H. PITTS, What the frog's eye tells the frog's brain, Proceedings IRE, 47, 1940 (1959).

12. A. L. ZOBRIST, A model of visual organization for the game of GO , Proceedings SJCC, 34, 101-110 (1969).

13. K. CRAIK, Hypothesis on the nature of thought, The Nature of Explanation, 50-61, Cambridge University Press, Cambridge (1943).