Algebraic inference of pattern similarity

By J. R. Ullmann*

It is common in automatic pattern recognition to recognize an unknown pattern as the same as the known pattern to which it is most similar. This paper explores the idea that if appropriate pairs of parts are chosen, then the similarity of a pair of patterns is the sum of the similarities of these pairs of parts. In a simplified recognition problem, it is found that appropriate pairs of parts are chosen automatically when a purely algebraic technique is used for inferring the similarities of unknown to known patterns. This is a critical step in a research programme aimed at automatically finding suitable systems of features for recognizing hand-printed and cursively written characters.

1. Introduction

Greanias, Meagher, Norman and Essinger (1963) have claimed 92% successful recognition of unconstrained hand printed numerals, in terms of intuitively chosen features such as "west bay, bottom left" (in recognizing "3" and "5") and "horizontal line end, top left" (in recognizing 2, 3, 7), etc. "West bay, bottom left" is not a single perfectly defined geometrical shape, and instead can be thought of as a collection of alternative exactly defined but slightly different shapes which count as "west bay, bottom left" and which we call *variants* of this feature. A "five" is not a single perfectly defined shape, but a class of fairly similar shapes which we call *variants* of this character.

Characters may be recognized in terms of properties more abstract than geometrical features. For example, Giuliano, Jones, Kimball, Meyer and Stein (1961) and Alt (1962) have obtained the higher moments of patterns, blackness being analogous to mass; and Horwitz and Shelton (1961) and Clowes and Parks (1961) have used autocorrelation as a first stage of automatic recognition. Generally speaking, the output of the first (preprocessor) stage is a set of numerical values. It is often found expedient to multiply these values by weights in the decision process, and several methods for determining suitable weights are now well known. Statistical weighting was proposed by Selfridge (1955) and implemented for example by Doyle (1960) on results of fairly complicated geometrical tests. Following Roberts (1960), Duda and Fossum (1966) have experimented with a perceptron-type systematic trial and error method for finding weights, and also for finding more than one set of weights per character.

An underlying presupposition in contemporary work is the *compactness* hypothesis. According to this it is indeed possible to find (preprocessor) tests which give numerical results when applied to raw patterns, and which have the following property. When the numerical results of these tests are used as the co-ordinates of a point in a hyperspace with a suitably chosen metric, the points corresponding to variants of the same character lie in a number of compact clusters, which are not entangled with the clusters representative of variants of other characters. Machines designed on this hypothesis have been fairly successful in recognizing multifont print when examples of the characters in all the fonts to be recognized have been available to the machine during a conditioning phase, or available in advance to the machine designer. A salient example of this success is the simulation by Liu (1964) in which characters were recognized in terms of n-tuples automatically chosen by a computer.

The problem of recognizing hand-printed characters is analogous to the problem of recognizing remarkably clearly machine-printed characters from an unlimited number of different fonts, of which only a few are known in advance. The compactness hypothesis has been far less successful when few, rather than all, of the fonts to be recognized have been known in advance. For recognizing hand-printed addresses on postal envelopes, for example, experimental results have not been satisfactory; and a method for automatically choosing suitable features or more abstract tests remains to be devised. Furthermore, the *a priori* chances of the compactness hypothesis being a practical basis for hand-written-character recognition are infinitesimal.

The present work is a preliminary investigation of a purely algebraic recognition process in which a raw pattern is not transformed into a set of numerical test results, but instead into a set of algebraic unknowns. In this process, suitable parts (feature variants) of patterns are found automatically. The main reason for this investigation is that the algebraic method offers scope for replacing the compactness hypothesis by a less restrictive hypothesis on which basis the recognition of unconstrained hand-printed and even cursive characters may eventually become practical. The computational problems involved are so formidable that the present paper is restricted to a simplified recognition problem related to practical character recognition as explained in Section 3.

2. The problem

For clarity the problem is formulated in terms of a machine M which operates on data generated by a

* *Post Office Research Station, Brook Road, Dollis Hill, London, N.W.2.*

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

machine G. G generates N-bit binary patterns on an ordered set, U, of N locations, in each of which 1 or 0 may be written. A *sub-pattern* is a pattern on a subset of U.

For use in the generation of patterns on U, certain constants are arbitrarily chosen by G but not revealed to M. These constants are a set

$$\Sigma = \{\sigma_1, \sigma_2, \ldots \sigma_i, \ldots \sigma_m\}$$

of positive integers, and a partition π_G which partitions U into the subsets

$$U_G = \{U_{1G}, U_{2G}, \ldots U_{iG}, \ldots U_{mG}\}$$

such that, for $i = 1, 2, \ldots m$, if λ_i is the number of elements in U_{iG}, then

$$2^{\lambda_i} \geqslant \sigma_i.$$

For each subset U_{iG}, for $i = 1, 2, \ldots m$, a set

$$V_{iG} = \{V_{iG1}, V_{iG2}, \ldots V_{iGj}, \ldots V_{iG\sigma_i}\}$$

of σ_i different alternative sub-patterns on U_i are randomly chosen. A different numerical value is assigned by G to each of these different sub-patterns, and this value is restricted to be a non-negative integer less than some arbitrary limit (but is otherwise randomly chosen).

From these chosen sub-patterns all possible patterns

$$V_\alpha = \{V_{1Gj(1)}, V_{2Gj(2)}, \ldots V_{iGj(i)}, \ldots V_{mGj(m)}\}$$

are assembled in turn, where all $j(i)$ are such that for $i = 1, 2, \ldots m$,

$$V_{iGj(i)} \epsilon V_{iG}.$$

Altogether there are

$$\psi = \sigma_1 \times \sigma_2 \times \ldots \times \sigma_m$$

such patterns, which are actually arranged by G in random sequence

$$V_1, V_2, \ldots V_\alpha, V_\beta, \ldots V_\psi.$$

For all α from 1 to ψ, V_α has a value which is the sum of the assigned values of its constituent sub-patterns, and G assigns V_α to one of the classes Q or R according as the value assigned to V_α does or does not exceed some arbitrary limit L_0.

This paper is concerned with a problem itself derived from the following simple recognition problem. Machine M is given as data the μ patterns $V_1, V_2, \ldots V_\mu$ from the set $V_1, V_2, \ldots V_\mu, \ldots V_\psi$, each labelled according to its membership of R or Q; and is also given that the value associated with any member of Q is greater than the value associated with any member of R. M is then given an unlabelled pattern V_ν from the set $V_{\mu+1}, \ldots V_\psi$, and is required to label it in agreement with G. It is important that M is not given Σ, π_G nor the actual values associated with the patterns or sub-patterns, and therefore this problem differs fundamentally from that treated for example by Ho and Kashyap (1965).

In the present problem M is implicitly given a number of inequalities of the form

(value for member of Q) $>$ (value for member of R)

and we call these the *data inequalities*. If it could be deduced from the data inequalities that for every given member of R

(value for V_ν) $>$ (value for given member of R),

then it would be at least fairly sensible to label V_ν as belonging to Q. A computational process for assigning truth values to inequalities such as

(value for V_ν) $>$ (value for given member of R)

could be tried out and tested on known inequalities such as

(value for member of Q) $>$ (value for member of R).

Therefore the present paper is concerned with the following problem:

Machine M is given as data the μ patterns $V_1, V_2, \ldots V_\mu$ from the set V_1. $V_2, \ldots V_\mu, \ldots V_\psi$, each labelled according to its membership of Q or R, and M is again also given that the value associated with any member of Q is greater than that associated with any member of R. M is then given patterns V_ϕ and V_ρ both belonging to $V_{\mu+1}, \ldots V_\psi$, such that $V_\phi \epsilon Q$ and $V_\rho \epsilon R$. M is required to perform a computation which assigns the truth value "true" to

(value for V_ϕ) $>$ (value for V_ρ),

and does not assign the value "true" to

(value for V_ρ) $>$ (value for V_ϕ).

As before, machine M is not given Σ, π_G, nor the actual values associated with the patterns of sub-patterns.

For example, if $N = 6$, M might be given

$$\begin{aligned} V_1 &= 110101, \quad V_1 \epsilon Q, \\ V_2 &= 001110, \quad V_2 \epsilon R, \\ V_3 &= 111000, \quad V_3 \epsilon R, \\ V_4 &= 011110, \quad V_4 \epsilon Q, \\ V_5 &= 111010, \quad V_5 \epsilon Q, \\ V_6 &= 010101, \quad V_6 \epsilon R, \\ V_\phi &= 111110 \\ V_\rho &= 001100 \end{aligned}$$

This example is considered in Sections 4 and 5.

3. Relationship to the practical problem of character recognition

Suppose that U comprises an even number of elements, and let U be divided into two non-overlapping sets, S and T, of $N/2$ elements. Further, let π_G be restricted so that in every subset of π_G there is at least one element of S and one element of T. We say that the intersections of

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

the same subset of U_G with S and T are *corresponding* subsets, and that sub-patterns on corresponding subsets are corresponding sub-patterns. Thus G assigns numerical values to pairs of corresponding sub-patterns.

It is common in practical character recognition to "recognize" a (perhaps preprocessed) unknown pattern as the same as the (perhaps preprocessed) stored pattern to which it is most similar. This calls for the computation of similarity of pairs of patterns, which is inevitably based on an hypothesis as to how the similarity of a pair of patterns is related to the patterns themselves. Let us consider the following hypothesis.

Partition Hypothesis. The similarity of a pair of patterns can be regarded as the value assigned to a pair of corresponding patterns one on S and one on T by a machine G, as described above. For this, π_G and the values assigned to pairs of corresponding sub-patterns must of course be properly chosen. According to this hypothesis, a pair of patterns can be partitioned into pairs of corresponding parts, such that the similarity of the pair of patterns is the sum of the similarities of the pairs of corresponding parts. This is an hypothetical generalization of the self-evident fact that a pair of identical patterns can be partitioned into pairs of identical corresponding parts.

If we make the simplifying assumption that any variant is more similar to any other variant of the same character than to any variant of any other character, then we have available, in practical character recognition, implicit data inequalities analogous to those introduced above in Section 2. For example, suppose that K_1 and K_2 are variants of the same character and K_3 and K_4 are variants of a different character. K_1 and K_2 can be regarded as patterns on S and T whose assigned value (their similarity) exceeds that of the patterns K_4 on S and K_2 on T. In practice many example variants are usually available, so that one has many inequalities of the form

(value for K_1 on S and K_2 on T) $>$
(value for K_4 on S and K_2 on T).

An unknown input pattern, K^{ν}, is to be recognized as the stored example to which it is most similar. The most similar stored pattern can be found if we can find for each and every pair of stored variants e.g. K_2, K_3, whether or not it is true that

(value K_{ν} on S and K_2 on T) $>$
(value for K_{ν} on S and K_3 on T).

Thus the recognition problem has been reduced to the problem of finding truth-values of inequalities, as in the problem of Section 2. The problem of finding π_G is analogous to the well-known problem of finding suitable features for pattern recognition. For example, on the partition hypothesis, it is conceivable that if a 5 were written on S and a 3 on T, the variants of "west bay, bottom left" in this 3 and 5 would be corresponding sub-patterns.

The partition hypothesis is of interest because of the prospect of generalization so that, for example, G is not confined to simple partitions, but instead to more general sets of subsets of U. It seems impossible to test the more general hypotheses in practice unless first an economical computational technique has been devised in solution to the problem of Section 2, viewed simply as a computational problem. The computational problems involved in more advanced hypotheses are presumably more difficult, quite apart form the question of validity of the hypotheses themselves. So the present paper is devoted to the problem of Section 2, and computer simulations, in the nature of pilot experiments, have been restricted to synthetic data.

4. Algebraic inference

The consistency of the equations

$$x + 2y = 8$$
$$2x + y = 7$$

implies the consistency of the equations

$$(a + b + c) + 2(a + b + d) = (3 + 3 + 2)$$
$$2(a + b + c) + (a + b + d) = (1 + 4 + 2),$$

where x is replaced by the same sum of arbitrary components in both equations, and the same is true for y. The set of components replacing x must differ in at least one component from the set replacing y. This elementary consideration is valuable in the solution to the problem of Section 2.

When G has assigned a value to a sub-pattern on a subset of U, that sub-pattern itself can be regarded as a name or symbol for that value. When we know the sub-pattern but do not know its assigned value, we can regard the sub-pattern itself as an algebraic unknown. (It is common to represent an algebraic unknown by "x". The symbol "x", written on paper, is itself a pattern or sub-pattern).

Just as, above, x was replaced by a sum of arbitrary components $a + b + c$, it is convenient now to replace a sub-pattern by a sum of arbitrary components, each of which is one of the digits of the sub-pattern, regarded as an algebraic unknown. The symbol "1" in a particular location in U does not generally signify or represent the value "one", but instead, some unknown value. A 1 in a given location in U is regarded as a different algebraic unknown, a different algebraic symbol, to a 1 in any other location in U, and the same is true for zeros.

Let us consider the example given at the end of Section 2. In this it is implicitly given that, for example,

$$110101 > 001110,$$

patterns being regarded as algebraic symbols for their associated values. The data inequalities also include, for example,

$$110101 > 111000$$

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

$$011110 > 111000$$
$$111010 > 010101$$

and the truth value of

$$111110 > 001100$$

is required.

Let the elements of U be u_1, u_2, u_3, u_4, u_5, and u_6, and let us for clarity rename, for example, a "1" in u_2 as "u_{21}", and a "0" in u_5 as "u_{50}". Using this notation and representing patterns by sums of components, the given inequalities may be rewritten

$$u_{11} + u_{21} + u_{30} + u_{41} + u_{50} + u_{61} > u_{10} + u_{20} + u_{31} + u_{41} + u_{51} + u_{60}$$
$$u_{11} + u_{21} + u_{30} + u_{41} + u_{50} + u_{61} > u_{11} + u_{21} + u_{31} + u_{40} + u_{50} + u_{60}$$
$$u_{10} + u_{21} + u_{31} + u_{41} + u_{51} + u_{60} > u_{11} + u_{21} + u_{31} + u_{40} + u_{50} + u_{60}$$
$$u_{11} + u_{21} + u_{31} + u_{40} + u_{51} + u_{60} > u_{10} + u_{21} + u_{30} + u_{41} + u_{50} + u_{61}.$$

Adding the first, third and fourth of these, and cancelling out some of the terms common to both sides of the sum we find

$$u_{11} + u_{21} + u_{31} + u_{41} + u_{51} + u_{60} > u_{10} + u_{20} + u_{31} + u_{41} + u_{50} + u_{60}$$

which establishes the required truth value.

The problem of finding rules of operation by which a machine can make such inferences is non-trivial because, for example, of the problem of deciding which inequalities to add together. It is not yet clear whether the rules of operation which we have actually used are heuristics for which no rigorous mathematical explanation can ever be given. But it is clear that the rules are essentially algebraic, in that they manipulate symbols for unknown values.

5. Program 1

(*a*) Notation and definitions

Different subsets, partitions, and patterns on U are distinguished by different lower case, upper case, and Greek subscripts respectively. Subset subscripts are such that the same subset has the same subscript whatever the partition on U in which it occurs. The range of an existentially quantified partition subscript is to be understood to be the range of all possible partition subscripts for a set of N elements. The range of an existentially quantified pattern subscript is to be understood to be the range of subscripts of all patterns stored in the computer. The range of a universally quantified subset subscript is to be understood to the range of all subset subscripts in the given partition. For example

(i) $(\exists B)(\exists \beta)(V_{iA\alpha} = V_{iB\beta})$

means "for all i such that $V_{iA\alpha} \epsilon V_{A\alpha}$, there exists a partition π_B on a pattern V_β, where β is in the range $1, 2, \ldots \mu, \phi, \rho$, such that the sub-patterns $V_{iA\alpha}$ and $V_{iB\beta}$ are the same".

For any two patterns V_α, V_β, where α, β are in the range $1, 2, \ldots \mu, \phi, \rho$, we define the proposition

$$H_{\alpha\beta} = ((V_\alpha \epsilon Q \cup R') \& (V_\beta \epsilon R \cup Q')) \vee ((V_\alpha \epsilon R \cup Q') \& (V_\beta \epsilon Q \cup R'))$$

where, initially, $Q' = \{V_\phi\}$ and $R' = \{V_\rho\}$.

We define F_0 as the set of all pattern partitions on $V_1, V_2, \ldots V_\mu, V_\phi, V_\rho$. For $n = 1, 2, 3, \ldots$, we define F_n by specifying that a pattern partition $V_{A\alpha}$ belongs to F_n if and only if

(i) $(\exists B)(\exists \beta)((V_{iA\alpha} = V_{iB\beta}) \& (\sim(\exists j)((V_{jA\alpha} = V_{jB\beta}) \& (j \neq i))) \& H_{\alpha\beta} \& (V_{B\beta} \epsilon F_{n-1}))$.

Thus every sub-pattern in $V_{A\alpha}$ must be the only sub-pattern common to $V_{A\alpha}$ and some other pattern partition belonging to F_{n-1} such that $H_{\alpha\beta}$ is true.

We define F_w as the first member of the series

$$F_0, F_1, F_2, \ldots F_n, \ldots F_{w-1}, F_w, \ldots$$

such that $F_w = F_{w-1}$.

(*b*) Computation

The members of $F_0, F_1, F_2, \ldots$ are found successively until F_w is reached. The program assigns the truth value "true" to the proposition

$$V_\phi > V_\rho$$

if and only if

$$(\exists A)(V_{A\phi} \epsilon F_w) \& (\exists B)(V_{B\rho} \epsilon F_w).$$

Otherwise no truth value is assigned to $V_\phi > V_\rho$. To test the reverse inequality, the members of F_w are found again, but now with $Q' = \{V_\rho\}$ and $R' = \{V_\phi\}$; and $V_\rho > V_\phi$ if and only if

$$(\exists A)(V_{A\phi} \epsilon F_w) \& (\exists B)(V_{B\rho} \epsilon F_w).$$

(*c*) Example

Let us consider the example given at the end of Section 2. Let the 203 partitions on the set $U = \{1, 2, 3, 4, 5, 6\}$ be

$$U_1 = \{1, 2; 3, 4; 5, 6\} = \{U_{11}, U_{21}, U_{31}\}$$
$$U_2 = \{1, 2; 3, 4, 5, 6\} = \{U_{12}, U_{42}\}$$
$$U_3 = \{1, 2, 5, 6; 3, 4\} = \{U_{53}, U_{23}\}$$
$$\vdots \qquad \vdots$$
$$U_{203} =$$

Where $Q' = \{V_\phi\}$ and $R' = \{V_\rho\}$, we find that F_1 includes many pattern partitions among which are

$$V_{21} = \{V_{113}, V_{426}\}$$

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

$$V_{22} = \{V_{11}, V_{424}\}$$
$$V_{13} = \{V_{121}, V_{235}, V_{31}\}$$
$$V_{24} = \{V_{126}, V_{422}\}$$
$$V_{35} = \{V_{53}, V_{213}\}$$
$$V_{26} = \{V_{124}, V_{421}\}$$
$$V_{3\phi} = \{V_{535}, V_{21}\}$$
$$V_{1\rho} = \{V_{122}, V_{23}, V_{313}\}.$$

Pattern partitions in this list depend on other members of this list for membership of $F_2, F_3, \ldots$ and in fact we find that $F_1 = F_w$, so that $V_\phi > V_\rho$ is taken to be true. In this example the chosen π_G was {1, 2, 4; 3, 4, 6}, and it is readily verified that $V_{G1}, V_{G2}, \ldots V_{G6}, V_{G\phi}, V_{G\rho}$ all belong to F_w.

When $Q' = \{V_\rho\}$ and $R' = \{V_\phi\}$, we find that F_1 includes, for example,

$$V_{2\rho} = \{V_{123}, V_{422}\}$$

but no partition on V_2 or V_3 belongs to F_1, so that $V_{2\rho}$ does not belong to F_2. In fact F_w is empty and therefore the proposition $V_\rho > V_\phi$ is not taken to be true.

(*d*) Economy

A disadvantage of program 1 is that it searches through all possible partitions on all given patterns, which is economically prohibitive if these patterns have many digits. The same difficulty arose in previous work which was concerned with equalities rather than inequalities. When it is given that the values associated with all generated patterns are equal, there is no point in distinguishing between Q, Q', R, R', and in fact "$H_{\alpha\beta}$" degenerates to "$\alpha \neq \beta$" in the condition for $V_{A\alpha} \epsilon F_n$, giving the condition that was actually used (Ullmann, 1966). A technique for achieving economy by *n*-tuple sampling (Ullmann, 1965) was found to work with the equalities program, which suggests that it would also work with the present inequalities program. But this sampling technique achieves economy only at the expense of error, and therefore an alternative method for avoiding working through all possible partitions has been developed and is presented below (program 2).

In order to run program 1 reasonably quickly, we have provided it with information as to the choice of π_G, which means that it has to search through fewer partitions. If π_G is completely given, program 1 needs only to search through all partitions on a set whose elements are the subsets of π_G, and the definition of F_n is unchanged. In the previous work on equalities, and in experiments 5 and 6 below, the program is given no information as to the choice of π_G.

6. Experiments with program 1

Experiments 1 to 4 were pilot experiments on eight-digit labelled patterns generated in accordance with Section 2. A number of labelled patterns were successively read by program 1 and simply stored as explained below. After this, when the τth pattern was read, if it belonged to Q it was taken as V_ϕ and the most recently stored member of R was removed from the list of stored Rs and taken as V_ρ. Alternatively, if the τth pattern belonged to R it was taken as V_ρ and the most recently stored member of Q was removed from the list of stored Qs and taken as V_ϕ. Program 1 then searched through $F_0, F_1, \ldots$ to find F_w. If the correct truth value was assigned to an inequality this was counted as *success* with the τth pattern, and if an inequality known to be false was assigned "true" this was counted as a *failure*.

In practical character recognition it would be absurd to store every pattern ever presented to the machine. Instead only sufficient patterns (or information abstracted from them) to allow correct recognition of all future patterns need be stored. So in experiments 1 to 4, patterns were stored according to the following rules.

If the first pattern to be read by program 1 was a Q it was stored in a list of Qs and if an R it was stored in a list of Rs. The same applied for subsequent input patterns until two members of Q and two members of R had been stored. For the next input pattern, program 1 worked through to find F_w. (When fewer patterns had been stored F_w would probably have been found empty.) For subsequent values of τ, the τth pattern was stored if labelled Q in the list of stored members of Q and if labelled R in the list of stored members of R only if success was *not* achieved with the τth pattern. Patterns which had been removed from the Q or R lists and taken as V_ϕ or V_ρ were returned to the lists whence they had been removed before the $(\tau + 1)$th pattern was read. A *run* was a sequence in which all patterns V_1 to V_ψ from the generator were read in turn and processed as has been described.

The programs were written in Elliott ALGOL: experiments 1 to 4 were run on an Elliott 503 and 5 and 6 on Atlas.

Experiment 1

The purpose of this experiment was to find how the choice of Σ in the generator affected the success rate of program 1, with π_G given. Ten runs were made for each choice of Σ, the choice of sub-patterns and their assigned values being different in each run (and actually derived from a pseudo-random number generator). With $\Sigma = \{2, 2, 2, 2\}$ a run took roughly one hour, and with $\Sigma = \{3, 2, 2\}$, roughly twenty minutes. The different rows of **Table 1** correspond to different Σ. The different columns show the numbers of successes at each τ from $\psi - 5$ to ψ in the course of ten runs. No failures occurred in this experiment.

From these results it appears that program 1 computes the truth value of a sufficient but not necessary condition for $V_\phi > V_\rho$, since this condition never fails but is not always succesful. It also appears that the success rate is higher the more subsets there are in Σ.

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

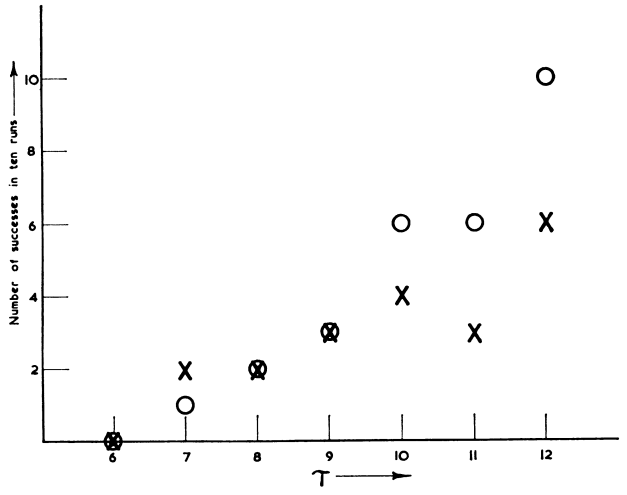

Fig. 1.—Performance with completely and incompletely specified *G*

Table 1

Performance of program 1

	τ					
Σ	ψ−5	ψ−4	ψ−3	ψ−2	ψ−1	ψ
{ 2, 2, 2 }	0	0	0	0	3	4
{ 3, 2, 2 }	2	2	3	4	3	6
{ 4, 2, 2 }	2	3	4	2	5	4
{ 4, 4 }	2	3	2	3	4	4
{ 3, 3, 2 }	4	5	3	3	7	6
{ 2, 2, 2, 2 }	6	9	9	9	6	8

Experiment 2

When π_G has three subsets 'known" to program 1, the set F_0 contains five partitions per pattern since there are five possible partitions on a set of three elements. If π_G were completely unknown to program 1 there would be 4,140 partitions per pattern in F_0, since this is the number of possible partitions on a set of eight elements. Experiment 2 investigated a compromise between these extremes, in which π_G was incompletely specified as follows.

The set U was partitioned into two equal sized parts S and T, which were partitioned into three pairs of corresponding subsets, as described in Section 3 above. The partitions on S and T were given to program 1, but which subset on S corresponded to which subset on T was left unspecified, and in fact F_0 contained 16 partitions per pattern. Σ was chosen to be $\{3, 2, 2\}$. In **Fig. 1** the circles give the numbers of successes at various values, counted over ten runs. There were no failures in this experiment The crosses are the corresponding $\Sigma = \{3, 2, 2\}$ results from experiment 1.

Applying the chi-squared test to these results we find

$$\chi^2 = 1{\cdot}09$$

whereas from tables

$$\chi^2_{2,\,5\%} = 5{\cdot}99$$

which indicates that the runs with incompletely specified π_G did not give significantly better results than when π_G was specified completely.

Experiment 3

In practical character recognition it is sometimes possible to find a character variant which intuitively seems more similar to a variant of a different character than to another variant of the same character. But a variant is most likely to be more similar to another variant of the same character rather than to a variant of a different character. This consideration led to the present experiment in which the first η patterns read by program 1 were deliberately mislabelled. For example, if the first pattern had been assigned by G to R, it now had its label changed to Q. Ten runs were made with $\eta = 1, 2, 3$, with $\Sigma = \{3, 2, 2\}$. The results for $\eta = 0$ are available from experiment 1. As explained above, program 1 did not attempt to find F_w unless a certain number of patterns had first been stored. **Fig. 2** shows the numbers of successes and failures per attempt per run, averaged over ten runs, for different η. This experiment indicates that the percentage drop in success is greater than the percentage of patterns mislabelled.

Experiment 4

One of the principal difficulties in the development of program 1 into a practical recognition technique is that the computation of F_w has to be repeated for every different pair, V_ϕ V_ρ. Purely sequential operation like this is so impracticable that it is of interest, even at this early stage, to see whether program 1 works when Q' and R' each have more than one member.

In experiment 1 F_w was found first for $Q' = \{V_\phi\}$, $R' = \{V_\rho\}$ and then for the reverse inequality with $Q' = \{V_\rho\}$ and $R' = \{V_\phi\}$. In experiment 4, F_w was found only once, with $Q' = \{V_\phi, V_\rho\}$ and $R' = \{V_\rho, V_\phi\}$, so that the inequality and reverse inequality were tested simultaneously. The inequality $V_\phi > V_\rho$ was taken to be true if V_ϕ in Q' and V_ρ in R' each had at least two partitions in F_w; and the reverse inequality was taken to be true in V_ρ in Q' and V_ϕ in R' each had at least two partitions in F_w. (Two partitions rather than one were required because pattern partitions induced by the trivial partition with only one subset on the members of Q' and R' inevitably belonged to F_w). Ten runs were performed with $\Sigma = \{3, 2, 2\}$ and π_G completely speci-

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

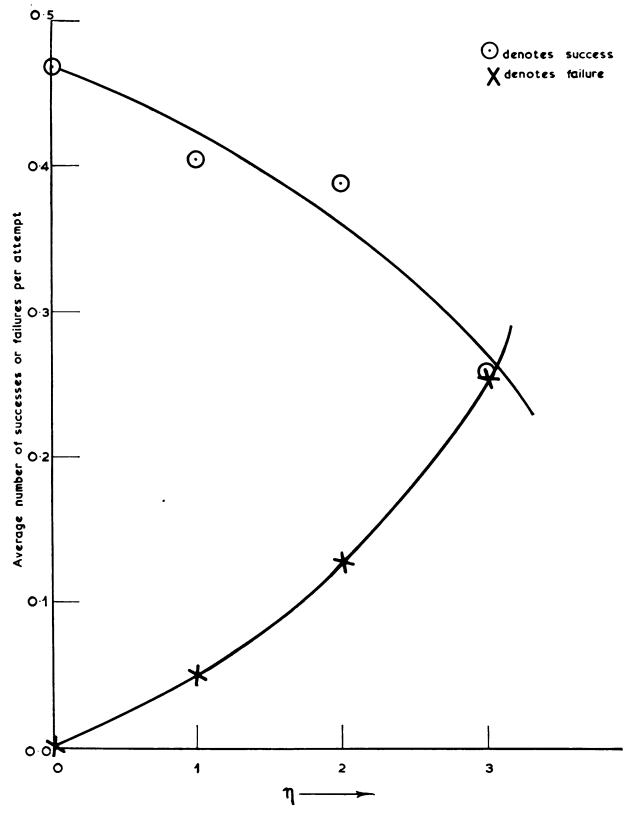

Fig. 2.—Success and failure rates with mislabelled patterns

fied, and **Table 2** compares the results with the corresponding experiment 1 results. This shows that the cost of speeding up the program by testing both inequalities simultaneously is an appreciable deterioration in performance.

7. Program 2

(*a*) Notation and definitions

Let Z be the trivial partition on U whose subsets are the single elements of U, so that, for example, U_{xZ} and U_{yZ} are single elements of U. Program 2 processes pairs of digits in which each digit belongs to a different pattern but is located in the same element of U. It is convenient to introduce a special notation in which $x\alpha\beta$ is the digit pair $\{V_{xZ\alpha}, V_{xZ\beta}\}$. This applies for any α, β from 1 to τ.

We define the sets C_0, C_1, C_2, ... C_n, ... and D_0, D_1, D_2 ... D_n, ... of digit pairs by means of the following conditions:

$x\alpha\beta\epsilon C_0$ if and only if $H_{\alpha\beta}$ & $(V_{xZ\alpha} \neq V_{xZ\beta})$

$x\alpha\beta\epsilon D_0$ if and only if $H_{\alpha\beta}$ & $(V_{xZ\alpha} = V_{xZ\beta})$

$x\alpha\beta\epsilon C_n$ if and only if

$H_{\alpha\beta}$ & $(\exists y)((y\alpha\beta\epsilon C_{n-1})$ & $(\exists \Upsilon)(\exists \delta)(\{x\alpha\Upsilon, y\alpha\Upsilon, x\beta\delta, y\beta\delta\} \subset D_{n-1}))$

$x\alpha\beta\epsilon D_n$ if and only if

Table 2

Comparison of performance in experiments 1 and 4

	SUCCESSES PER ATTEMPT		FAILURES PER ATTEMPT	
	AVERAGE	S.D. OF AVERAGE	AVERAGE	S.D. OF AVERAGE
experiment 4	0·44	0·43	0·13	0·20
experiment 1	0·47	0·51	0·00	0·00

$(x\alpha\beta\epsilon D_{n-1})$ & $(y)((y\alpha\beta\epsilon D_{n-1}) \vee ((y\alpha\beta\epsilon C_n)$ &

$(\exists \Upsilon)(\exists \delta)((\{y\alpha\Upsilon, y\beta\delta\} \subset D_{n-1})$ & $(\{x\alpha\Upsilon, x\beta\delta\} \subset C_n))))$.

We define D_w as the first member of the series D_0, D_1, ... D_n, ... D_{w-1}, D_w, ... such that $D_w = D_{w-1}$.

(*b*) Computation

Program 2 finds successively the members of C_0, C_1, C_2, ... D_0, D_1, D_2, ... until D_w is reached. The program assigns the value "true" to the inequality $V_\phi > V_\rho$ if and only if every digit in V_ϕ and V_ρ belongs to at least one digit pair belong to D_w. The same criterion is applied when D_w is found again with $Q' = \{V_\rho\}$ and $R' = \{V_\phi\}$ to test the reverse inequality.

(*c*) A note on the relationship between programs 1 and 2

For the purpose of this perfunctory comparison, let us define C' and D' as sets of pairs of digits (the element of U being the same for both members of each pair), belonging to not-identical and identical sub-patterns respectively. Let us consider the condition given in Section 5(*b*) for $V_{A\alpha}\epsilon F_n$, and let us consider a digit $V_{xZ\alpha}$ in $V_{iA\alpha}$, where $V_{iA\alpha}$ is the only sub-pattern common to $V_{A\alpha}$ and $V_{B\beta}$. Any other digit $V_{yZ\alpha}$ must also belong to $V_{iA\alpha}$ in which case

$$y\alpha\beta\epsilon D' \qquad \text{(i)}$$

or $V_{yZ\alpha}$ must belong to some other subset $V_{jA\alpha}$, in which case

$$(\exists E)(\exists \Upsilon)(V_{jA\alpha} = V_{jE\Upsilon})$$

and thus

$$y\alpha\Upsilon\epsilon D'. \qquad \text{(ii)}$$

It is also necessary that if $V_{B\beta}$ includes a sub-pattern $V_{jB\beta}$, then $V_{jB\beta} \neq V_{jA\alpha}$, since $V_{iA\alpha}$ is to be the only common sub-pattern, and thus

$$x\alpha\beta\epsilon C'. \qquad \text{(iii)}$$

Furthermore, $V_{jE\Upsilon}$ is to be the only common sub-pattern between $V_{A\alpha}$ and $V_{E\Upsilon}$, so that since $V_{xZ\alpha}$ is overlapped by $V_{iA\alpha}$, it follows that

$$x\alpha\Upsilon\epsilon C'. \qquad \text{(iv)}$$

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

Combining (i), (ii), (iii), (iv) we obtain the insufficient condition $x\alpha\beta\epsilon D'$ if

$$(y)((y\alpha\beta\epsilon D') \vee ((y\alpha\beta\epsilon C') \,\&\, (\exists \Upsilon)((y\alpha\Upsilon\epsilon D') \,\&\, (x\alpha\Upsilon\epsilon C'))\rangle\rangle.$$

The analogy between this and the condition for $x\alpha\beta\epsilon D_n$ is sufficiently obvious to throw a little light on the relationship between programs 1 and 2.

8. Experiments with program 2

Experiment 5

Before D_n was computed, it was required only that at least one member of Q and one member of R had been stored. In this experiment, runs were made with two different values of N, the number of elements in U. Otherwise experiment 5 differed from experiment 1 only in the use of program 2 rather than program 1. **Table 3** shows the number of successes with $\tau = \psi - 5, \psi - 4, \ldots, \psi$. There were no failures in this experiment, and π_G was completely unspecified. The results indicate that the performance of program 2 is similar to that of program 1.

Experiment 6

In Section 3 it was pointed out that the partition hypothesis is an hypothetical generalization of the fact that identical patterns may be partitioned into corresponding pairs of identical parts. *Whatever* the partition on one pattern there exists an identical partition on an identical pattern, such that the pairs of corresponding subpatterns are identical. Generalizing this, we do not expect to find just one partition such that the similarity of two patterns is the sum of the similarities of the parts, but rather, a whole class of partitions for which this is so. For simplicity, the machine G had hitherto been confined to a single partition π_G, but in experiment 6 it used two partitions, π_G and π_H. (To use more than two partitions in G, or to use larger N, it would have been necessary to combine program 2 with an n-tuple economy technique (Ullmann, 1965), in order to overcome storage space limitations. But the added complexity seemed unwarranted at this stage.)

Table 3

Performance of program 2

		τ					
N	Σ	$\psi-5$	$\psi-4$	$\psi-3$	$\psi-2$	$\psi-1$	ψ
4	{ 2, 2, 2 }	0	0	0	4	5	7
4	{ 3, 2, 2 }	1	2	5	4	2	3
8	{ 3, 2, 2 }	2	4	4	3	8	7

In experiment 6, the generator G generated patterns V_{1G} to $V_{\psi G}$ and labelled them exactly as in Section 2. It then generated another ψ patterns V_{1H} to $V_{\psi H}$ in the same way, but based on π_H rather than π_G, with labels assigned as before. If any pattern in V_{1H} to $V_{\psi H}$ was the same as any pattern in V_{1G} to $V_{\psi G}$, the patterns V_{1H} to $V_{\psi H}$ were rejected and replaced by a newly generated set. The labelled patterns V_{1G} to $V_{\psi G}$ and V_{1H} to $V_{\psi H}$ were then shuffled up into random sequence and presented to program 2 as a run of 2ψ labelled patterns, the experimental procedure being otherwise the same as in experiment 5. Program 2 itself was not modified, and ten runs were made, with $N = \sigma$ and $\Sigma = \{2, 2, 2\}$ giving the following results:

Average number of failures per run 1·1

Average number of successes per run 2·6.

This shows that the introduction of a second partition causes deterioration of performance.

Acknowledgement

Acknowledgement is made to the Engineer-in-Chief of the General Post Office for permission to publish this paper.

References

ALT, F. M. (1962). Digital Pattern Recognition by Moments, in *Optical Character Recognition* edited by Fischer, Pollock, Raddack and Stephens, Spartan Books, pp. 153–179.

CLOWES, M. B., and PARKS, J. R. (1961). A New Technique in Automatic Character Recognition, *The Computer Journal*, Vol. 4, pp. 121–128.

DOYLE, W. (1960). Recognition of Sloppy Hand-Printed Characters, *Proceedings of the* 1960 *Western Joint Computer Conference*, Vol. 17, pp. 133–142.

DUDA, R. O., and FOSSUM, H. (1966). Pattern Classification by Iteratively Determined Linear and Piecewise Linear Discriminant Functions, *Transactions of the IEEE on Electronic Computers*, Vol. EC 15, No. 2, pp. 220–232.

GIULIANO, V. E., JONES, P. E., KIMBALL, G. E., MEYER, R. F., and STEIN, B. A. (1961). Automatic Pattern Recognition by a Gestalt Method, *Information and Control*, Vol. 4, pp. 332–345.

GREANIAS, E. C., MEAGHER, P. F., NORMAN, R. J., and ESSINGER, P. (1963). The Recognition of Numerals by Contour Analysis, *IBM Journal*, Vol. 7, No. 1, January 1963, pp. 14–21.

HO, Y-C., and KASHYAP, R. L. (1965). An Algorithm for Linear Inequalities and its Applications, *Transactions of the IEEE on Electronic Computers*, Vol. EC14, No. 5, pp. 683–688.

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024

HORWITZ, L. P., and SHELTON, G. L. (1961). Pattern Recognition Using Autocorrelation, *Proc. IRE*, Vol. 49, No. 1, pp. 175–185.

LIU, C. N. (1964). A Programmed Algorithm for Designing Multifont Character Recognition Logic, *Transactions IEEE*, Vol. EC13, No. 5, pp. 586–593.

ROBERTS, L. G. (1960). Pattern Recognition with an Adaptive Network, *IRE International Convention Record*, Pt. 2, pp. 66–70.

SELFRIDGE, O. G. (1955). Pattern Recognition and Modern Computers, *Proceedings of the* 1955 *Western Joint Computer Conference*, pp. 91–93.

ULLMANN, J. R. (1965). A Basic Approach to Pattern Recognition, *The Computer Journal*, Vol. 7, No. 4, pp. 282–289.

ULLMANN, J. R. (1966). Associating Parts of Patterns, *Information and Control*, Vol. 9, No. 6, pp. 583–601.

Book Review

Theory of Self-Reproducing Automata, by J. VON NEUMANN, edited and completed by A. W. BURKS. 1966; 388 pages. (*University of Illinois Press*, 75s.)

This volume contains von Neumann's far-reaching ideas on automata. Scientists will be thankful to Arthur Burks for presenting von Neumann's lectures and manuscripts, so making his last work on computer science available to the public.

The breadth of von Neumann's contribution to computer science, including automata theory, is well summarized in the editor's introduction. Von Neumann was particularly interested in large-scale natural (nervous systems) and artificial automata. He worked towards a theory of the logical organization of complicated systems of computing elements and believed that such a theory was an essential prerequisite to the construction of very large-scale computers. The two problems on which he worked in detail—reliability and self-reproduction—are both related to problems of complexity. Because of his premature death in 1957 it was left to Arthur Burks to put von Neumann's research on automata into final form.

Part I of this work is an edited version of a recording of five lectures on "Theory and Organization of Complicated Automata" delivered by von Neumann at the University of Illinois in 1949. Part II is an edited version of an unfinished manuscript entitled "The Theory of Automata: Construction, Reproduction, Homogeneity," which von Neumann started in 1952 and worked on for a year. It is largely concerned with the problem of self-reproduction of automata. Von Neumann's manuscript is incomplete: in Chapter 5 Burks completes the design of a self-reproducing automaton.

A fundamental question in the theory of automata is "when is a class of automata logically universal; is any single automaton logically universal"? Von Neumann recognized that Turing had provided an answer to this question by showing that any logical process (computation) that can be effected by finite but arbitrarily extensive means can be performed on a Turing machine and that there is a single such machine that can perform any given computation. Von Neumann raised and answered analogous questions concerning construction:

(*a*) Can an automaton be constructed by another automaton? What class of automata can be constructed by a single suitable automaton?

(*b*) Is any single automaton construction-universal?

(*c*) Is there a self-reproducing automaton?

In developing the answers to these questions von Neumann considered two methods of self-reproduction. The "kinematic model" is discussed in Part I (the Illinois lectures). Part II discusses the "cellular model", a conception stimulated by Ulam. In this model, reproduction occurs in an indefinitely large space regularly divided into cells, each of which contains a copy of a single finite automaton. Von Neumann chose for detailed development an infinite two-dimensional array of square cells each of which contains a copy of a certain 29-state finite automaton. The structure of this automaton is given in Chapter II by defining its "transition rule", the function which gives its state at time $t + 1$ in terms of its own state and those of its four contiguous neighbours at time t. The temporal reference frame for the cellular structure consists of instants $\ldots -2, -1, 0, 1, 2 \ldots$ All cells are in the same quiescent state at negative times. At time zero this total homogeneity is perturbed by changing the states of the automata within a bounded area A of cells. This homogeneity will, in general, propagate into the surrounding area, the course of this activity being uniquely determined by the transition rule of the automata and the conditions imposed at $t = 0$. In the case of self-reproduction the inhomogeneity of area A spreads until at time t and area A', disjoint from A, is so organized that the states of automata in A' and in A at time t are identical to the corresponding states in A at 0.

Von Neumann constructs self-reproduction as follows. An automaton, abstractly conceived, consists of a finite-state mechanism reading and writing on an infinite tape. Such a system can be embedded in the 29-state cellular structure by reserving an infinite array L of cells to act as the tape, the finite state mechanism being realized by a certain assignment of states to a finite set of cells adjoining L. In Part II is shown in detail the embedding in the cellular structure of a certain automaton M_c, analogous to a universal Turing machine, with the property: for each initially quiescent automaton M there is a description $D(M)$ of M (coded in terms of states of cells) such that when $D(M)$ is placed on L, M_c will construct M. Thus M_c is universal in the sense of question (*b*). Question (*c*) is reduced to (*b*) by showing how M_c can reproduce itself. In essence this is accomplished by placing a description of M_c itself on L.

J. P. CLEAVE (Bristol)

Downloaded from https://academic.oup.com/comjnl/article/10/3/256/494029 by guest on 13 March 2024