A Reprint from

INFORMATION THEORY

THIRD LONDON SYMPOSIUM

Papers read at a Symposium on 'Information Theory'
held at the Royal Institution, London,
September 12*th to* 16*th* 1955

Edited by
COLIN CHERRY

Published by
BUTTERWORTHS SCIENTIFIC PUBLICATIONS
88 KINGSWAY, LONDON, W.C.2

33

PATTERN RECOGNITION AND LEARNING*

OLIVER G. SELFRIDGE

Massachusetts Institute of Technology, U.S.A.

MANY psychologists studying learning have assumed that the subject—rat, dog, or graduate student—invariably knows what the stimulus is. They have not concerned themselves with how a dog knows that it is the bell ringing which is the stimulus to jump over a fence. A bell ringing never gives the same set of nervous impulses into the brain twice (of course the argument would still apply even if it did); why then should the dog classify all cases of bell ringing into one category—'stimulus'? There is then the further question of how this category is more or less quickly 'associated' with a response: the point is that the stimulus is not *a priori* considered a significant entity by the subject.

In designing programmes for computers to imitate conditioned reflexes, for instance, we have found that the real problem was to identify the stimulus. It was very easy to have sigmoid curves of facilitation and inhibition and so on, but the ringing of a bell, for example, is not a single, unique input. No ringing bell sounds exactly the same twice, in the sense that the entire waveforms are identical. How then is the machine to know a bell from a buzzer?

From an analytic point of view a pattern is merely a subset of all ensembles of a field of data; that is, if every datum has all possible values, every ensemble of those data does or does not belong to that pattern. Accordingly, every visual image either is or is not, say, a circle, and either is or is not a square (note that it is not necessarily inherently determined that the two classes do not overlap). Every sound, similarly, either is or is not a bell ringing, and so on.

It is no essential restriction on the data to consider that every datum is 0 or 1. Hence, every pattern is equivalent to some logical function on the field of data, such that every example of the pattern is valued 1 and everything else 0. This is, of course, to some extent valid; what is not valid is the converse, that any logical function can be a pattern.

It is submitted that the analytic notion of pattern is unreasonable; rather I would give an operational definition—a pattern is equivalent to a set of rules for recognizing it.

If the number of data, on which the pattern ought to be a logical function, is very large the rules for recognizing the pattern will probably not always give an unequivocal answer. There are, for example, bearded men and clean-shaven men, and off-hand one might suppose that these classes

* This paper has been slightly reduced from the original—Ed.

were not only exclusive, but inclusive. In fact, of course, very many people are neither bearded nor clean-shaven. It is true that one's definition of 'bearded' can become more and more precise so that fewer and fewer people are left in the middle category, but it is precisely when one does this that other people disagree with one's classification. The point is that a pattern is determined by the rules for recognizing it rather than *vice versa*, and the redundancy of the world is not always tailor-made to the language in which the rules must be stated.

Pattern recognition versus learning

One of the most impressive things about the brain is the great amount of 'information' that flows into it, and it is very clear that these incoming data are vastly redundant. Under certain conditions, sets of these data are categorized into classes or *patterns*, the number of which is very much smaller than the entire possible class of logical functions of the inputs. This implies that the data have been filtered and that the number of bits to be handled is now much decreased.

From the point of view of synthesis we can say that the categories are far from purely arbitrary functions on the space of all sets of data inputs. Each category must be invariant under some of certain commonly met transformations. For example, faces as visual patterns are subject to magnification, translation, intensification, blurring and rotation, and they remain the same faces still. Every pattern has its own invariances, and for each one of these it has some range of invariance; 'in focus' has only a slight range of blurring.

A 'square' cannot be compressed in one direction very much without metamorphosing into a rectangle or parallelogram (and *not* a square); if it is rotated 45° it becomes a 'diamond' for many people, but as long as it has four corners, in roughly the right places, it is a recognizable square for most people.

By 'pattern recognition' is meant classifying a set of data into the learnt categories; by 'learning' acquiring feasible operational definitions of the categories. Thus 'learning' and 'pattern recognition' are complementary.

Visual pattern recognition—the model

Figure 1 shows a block diagram of a model. An original image (90 × 90 0's and 1's) is transformed into a secondary image by one of the three operations A, B, and C; the secondary image itself may be transformed a number of times. After the image has been transformed by this sequence of operations, the 1's left in the image are counted. A typical original image is transformed sequentially in *Figure 2*, showing the secondary images at each step. The final count is then compared with the numbers stored for that particular sequence under the various symbols. If, after a number of sequences have been run on an image, the counts check sufficiently well with the stored distributions of symbol 1, say, the computer may identify that image as symbol 1.

We do not pretend that our operations are a complete set, but hope that they are comprehensive enough to prove our point. There are three kinds,

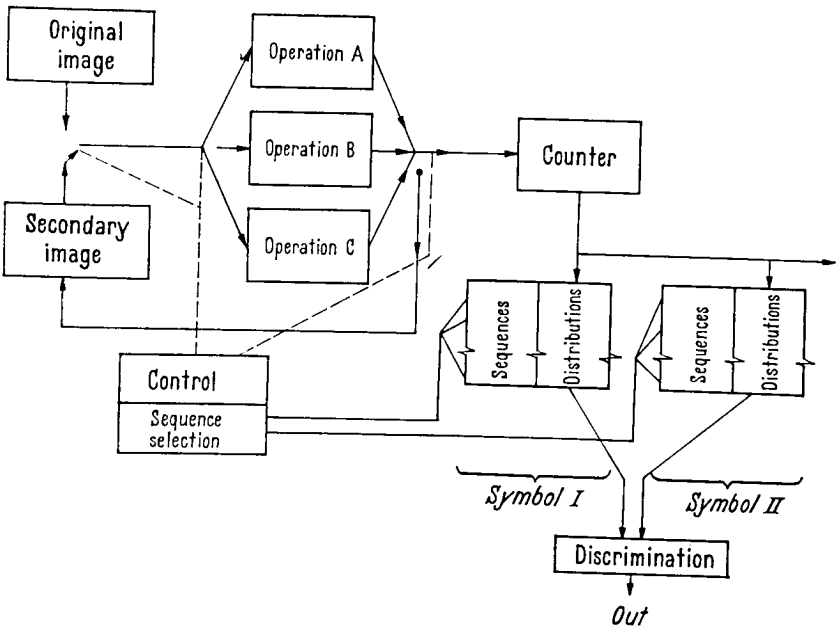

Figure 1.

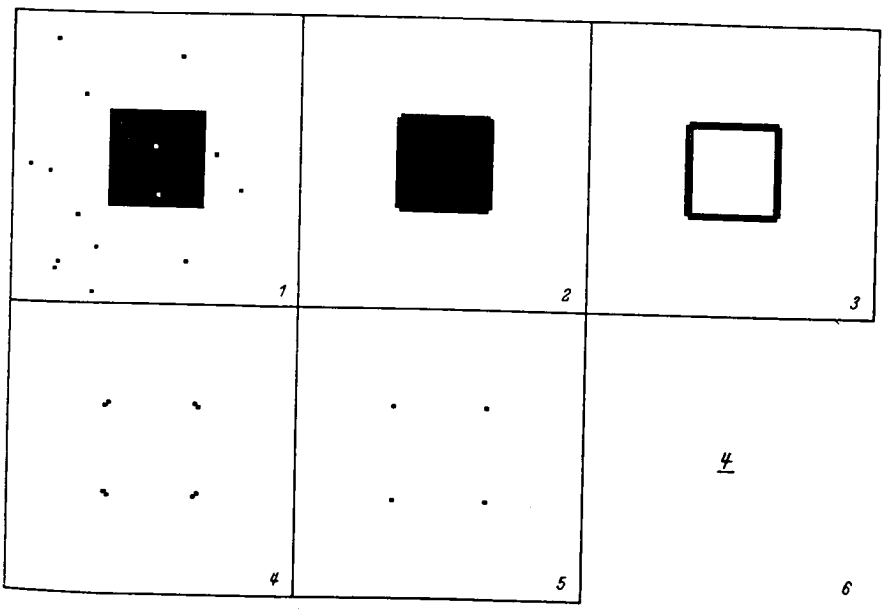

Figure 2. Image processing

and they are all local and isotropic.* The first, local averaging, replaces each datum by an average of the neighbouring data; thus, *inter alia*, it eliminates granular noise, isolated 1's in a field of 0's, and isolated 0's in a field of 1's, emphasizing local homogeneity. The second operation, local differencing, replaces each datum by an average of the logical differences

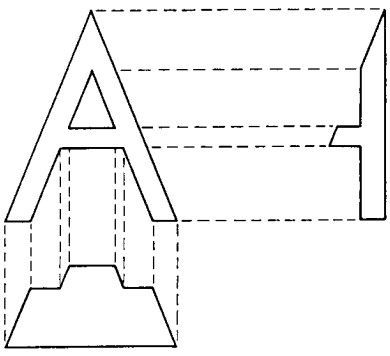

Figure 3. Projection as a means of reducing a two-dimensional image to a one-dimensional image.

of the neighbouring data, thus emphasizing local discontinuities and tending to sharpen contrasts, edges and corners. The third operation, 'blobbing', replaces each relatively isolated conglomeration of 1's by a single 1.

Many kinds of visual features cannot be handled by our particular operations. For example, this model cannot distinguish a capital C from a capital U (being a mere 90° rotation).

We deliberately restricted the kinds of things our model could recognize because the digital computer was, like them all, slow and small; even though it had 65,000 bits accessible within 10 μsec, and five times that many in slightly longer, it still took as long as 15 minutes to process an image. We have seen that our operations are powerful enough to detect and count critical points, and even to compute some measure of curvature. Furthermore one can project along any direction, which will reduce a two-dimensional set of data to a one-dimensional one (see *Figure 3*); or again one can 'thin', that is, change, the exterior 1's of an image to 0's so long as one does not alter the topological connectivity. It is not necessary, either, that the final data reduction be counting; one might store the topological connectivity instead.

It is not the particular sequences of operations it uses which makes our machine instructive but the way in which it hunts for good sequences. Every sequence is good or bad according as the numbers obtained by applying two images tend to differ consistently for different symbols. The essence of learning, it seemed to us, lies in having the computer recognize the pattern of good sequences. At the beginning it selects sequences in a random fashion, but as soon as it finds some of the sequences better than others it fashions and tests new sequences like the successful ones. The concept of similarity of sequences is built in, governed merely by a matrix

* This seems to reflect a universal prejudice of nervous systems.

of transition frequencies. All the sequences to be tested are fashioned (*cf.* Monte Carlo) by this matrix of transition frequencies, which is initially flat. As soon as a successful sequence appears its transition frequencies are used to bias that matrix, which then represents a hypothesis about (the pattern of) good sequences. This hypothesis can be tested by using

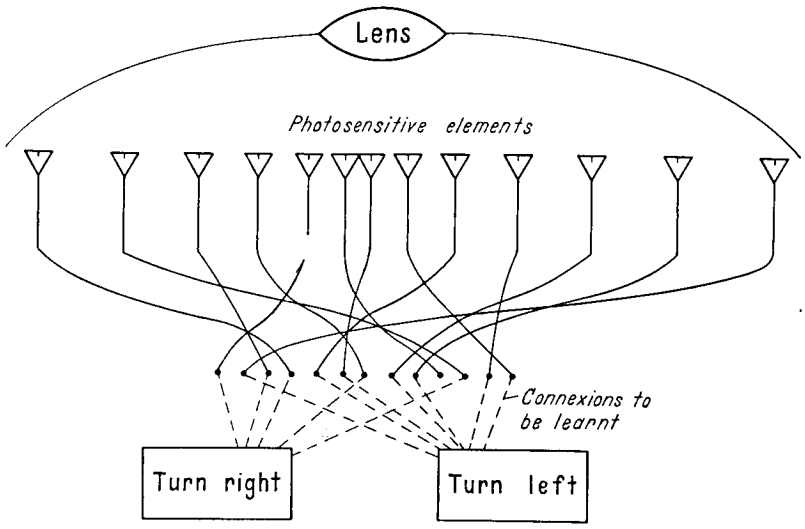

Figure 4. Organizing a one-dimensional visual field.

it to build new sequences and discarding or accepting it and them according as they prove useless or useful.

We do not maintain that this method is necessarily a very good way of choosing sequences, but it should be more successful than one which ignores what kinds of sequences have worked.

Learning—the general case

The machine is presumably maximizing, as well as it can, some function of its inputs, which we may call 'hedony' or pleasure. The only control it has over its inputs derives from observing correlations of its actions and the world's reaction to them. The machine may start off with as much or as little *a priori* knowledge of those correlations as we want.

From the atomic data inputs the machine derives its first testable guesses or inductions by means of whatever built-in data processing devices it has. They may be, for example, low order correlations of a sequence of data. Any inductions that prove useful or helpful can be considered as atomic inputs on a one-higher level of abstraction; about these, too, guesses may be made and tested, which may themselves be subject to analysis and induction.

Learning, as I see it, is thus an arboriform stratification of guesses about the world. New guesses, or new concepts (new patterns), are essentially simple combinations of words which form the patterns that have already been learned or that are inherent.

In a large sense, learning is the evolution of patterns; and the first steps are the hardest. The first word takes the longest time to learn, and very few people have won their first game of chess. Not only this, but small steps are easy and marginally profitable, while the big ones are hugely profitable and almost impossible to make.

Primitive data organization

We exhibit here an example of the development of organization in visual data (while the elements of our model might be construed as pseudo-neurons, they are not intended to be). It is difficult to find natural problems for synthesis on this level, because here the computing machine designer is most sure of himself. He cannot yet adequately define visual patterns, so that some degree of learning must be inserted into the programmes; he knows so little about complicated patterns like chess that almost the whole problem is learning. But he knows the topology of the plane. It is also true that a machine at this level of organization has very little stratification of its concepts, so that progress is slow and the learning primitive.

Our machine has a number of inputs from an optical system and a number of photosensitive elements (see *Figure 4*) which are most heavily concentrated at the centre. The *purpose* of the machine is to maximize the total intensity I of its inputs. To control this it has two outputs, left and right, which can rotate the 'eye' left and right respectively. The problem is how to assign the inputs to the proper outputs so that in fact I is maximized.

Consider time quantized. We start projecting stable images, one at a time, into the 'eye'. After any period we may compute the success of the machine by calculating the increase or decrease in I. Then we make the probability of altering the connexion of any input element to one of the two output elements dependent on its history. Thus, if some element has usually contributed to the wrong output (noting that I is negative) we should raise the probability of altering its connexion and *vice versa*.

FARLEY and CLARK[1] simulated on a digital computer* a situation rather like this and found that the process did in fact converge on to the right connexions. There are additional analytic grounds for believing that such a machine should converge on to optimum behaviour: for consider a small animal, senseless save for an altimeter, who makes random steps in any direction, but whose probability of standing still varies directly with the altitude. It is trivial to show that his net mean velocity *up* hill is positive and varies directly with the slope†.

It is interesting to note that if the photosensitive units in our model respond to changes in intensity instead of merely intensity (*e.g.*, if they are capacity-coupled to the input leads) the 'eye' will tend to fix on moving objects.

The chess-learning machine

For some time now people[2, 3] have been talking about machines learning to play games like chess. If we mean by this that a machine with no *a priori*

* At Lincoln Laboratory, M.I.T.

† The time to climb such a hill varies inversely with the square of the slope. It makes no difference how many dimensions there are in the space our creature is wandering in.

knowledge of the game save of its rules learns to play it passably well (and perhaps not even that) then I submit that no adequate design for such a machine has ever been demonstrated. I do not intend to do so here.

I do not wish to imply that the machines proposed are frivolous, but that they do not solve the critical problem. Many of these proposals have the

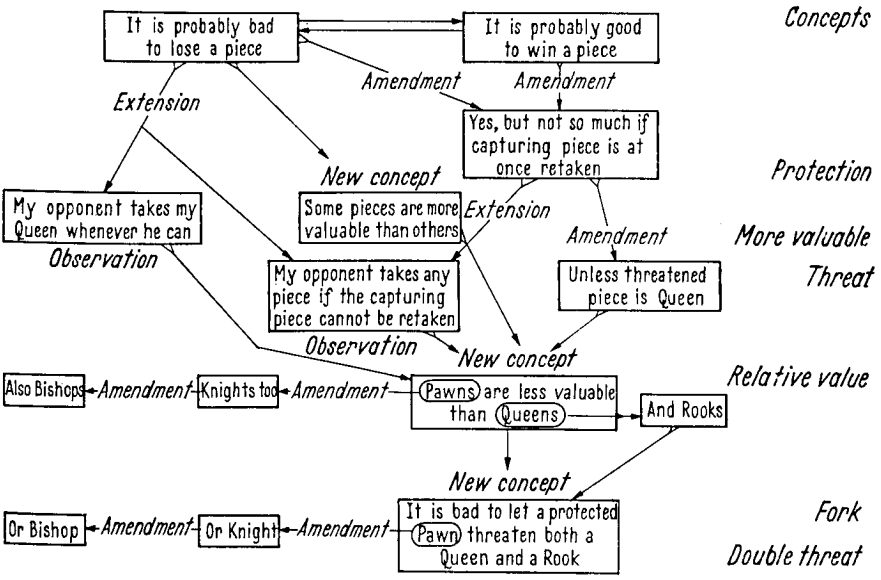

Figure 5. A possible small section of a growing set of rules of thumb. Blocks at butts chronologically precede blocks at heads.

machine look ahead some number of moves and select the move on the basis of the results of that inspection; but surely, unless the position is such that one side is close to mate, that process must involve evaluating the positions reached by those moves, and if the machine can evaluate positions why does it bother to look ahead more than one move? It would be better if the machine could evaluate all the positions after every possible move on its part and choose the move which maximizes the evaluation. The critical problem is clearly to acquire a better and better evaluation function.

I suggest that this usually occurs in people by their first formulating simple rules which do not even pretend to be evaluation functions. They are extended and amended, they are corrected or proved false, they are balanced against one another, and so on, until they do in fact together form implicitly a workable evaluation function.

Figure 5 illustrates a very small section of a possible chain of such rule-of-thumb inductions and observations; *Figure 6* shows another. Beside the inductions are the names of some of the concepts implicit in them.

At the beginning of any hypothetical chess-playing machine's history it cannot be said that two chess positions are similar unless, perhaps, its descriptive statements of them are largely identical, because, as before,

'similar' is taken with respect to the patterns that have already been formulated, that is, with respect to the descriptive language derived from practice. As soon as rule induction has proceeded enough to enlarge the initially necessarily sterile descriptive language of position, more and more hypotheses can be considered, tested, and accepted or rejected. It may then recognize two similar features in two different positions.

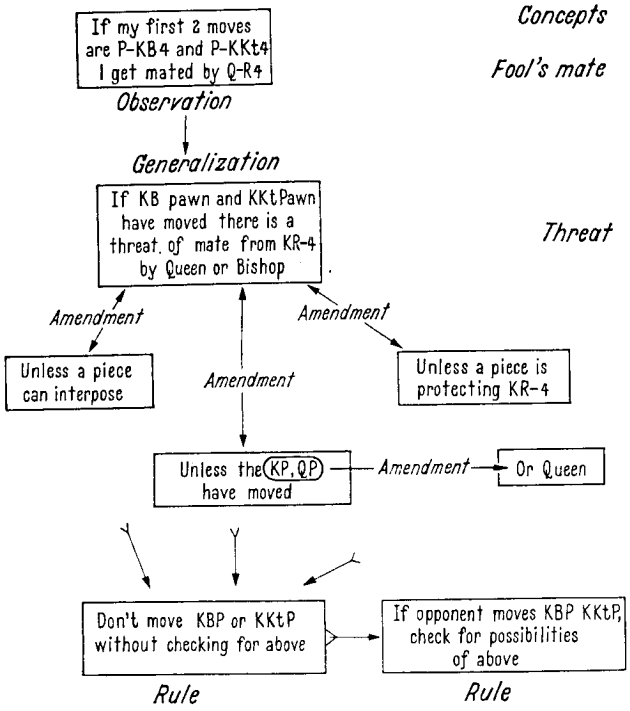

Figure 6. Small part of set of rules of thumb. These concern the opening moves.

How does the machine then construct hypotheses about the game that it can test? This process is very similar to the other examples we have considered, and consists of combining, in some simple way, words and phrases and propositions which are already in use. Useful combinations that work significantly in practice can be given new shorter names (concept formation).

One trouble with chess, from the machine's point of view, is that it is a man-made game for men. Certain concepts arise from the game, words that we use to describe positions, 'value', 'double threat', 'the centre' *etc*; and we start learning to play chess with these concepts already formed, because we use them elsewhere. This, of course, gives us an enormous advantage over any machine without them, or which has to induce them from chess, where, after all, they possess some degree of subtlety.

I suggest at this point that there are two instructive ways to overcome the machine's want of a useful descriptive language. One could provide the machine *a priori* with a certain set of aphorisms like 'other things being equal, it is bad to lose a piece'. Alternatively, one could teach the machine

the language useful in chess by teaching it other simpler games. The second is, naturally, the course followed by man*.

The one exception to the general statement I made about the inadequacy of all designs for chess-learning machines is a paper by NEWELL[4]. He goes rather thoroughly into the structure of a language that the machine must have to talk about chess and into the kind of decisions and goals it must have; but he does not specifically discuss learning in this paper although 'learning considerations have been prominent in the thinking and motivation behind the machine'.

* It is true that some people learn to play chess by reading books: this is perhaps equivalent to building in a good evaluation function.

REFERENCES

[1] FARLEY, B. G. and CLARK, W. A. 'Simulation of Self-Organizing Systems by a Digital Computer', *Proceedings Western Joint Computer Conference, March 1955*, I.R.E., A.I.E.E., A.C.M. in press

[2] SHANNON, C. 'Programming a Computer for Playing Chess', *Phil. Mag.*, 41 (1950) 256–275

[3] WIENER, N. *The Human Use of Human Beings* pp. 203–206. Boston; Houghton-Mifflin Company, 1950

[4] NEWELL, A. 'A Chess-Playing Machine', *Proceedings Western Joint Computer Conference, March 1955*, I.R.E., A.I.E.E., A.C.M. in press

[5] DINNEEN, G. P. 'Programming Pattern Recognition', *Proceedings Western Joint Computer Conference, March 1955*, I.R.E., A.I.E.E., A.C.M. in press

Made in Great Britain at the Pitman Press, Bath

Bruce Buchanan <bbuchana@pitt.edu>

To: Edwina Rissland

do you have and could you loan ...

the 1956 London Symposium volume? I'd send it out to be digitized so we have this paper and return it with no damage. (If I can't get permission to put up the whole volume I'll just do this paper.)

thanks,
bgb

(Selfridge 1956) Oliver G. Selfridge. Pattern recognition and learning. In Colin Cherry, editor, Proceedings of the Third London Symposium on Information Theory, New York, New York, 1956. Academic Press.