6. References

[1] LYAPUNOV, A. A.: *Basic Problems of Cybernetics.* Collection Problems of Cybernetics. Moscow: 1958.

[2] PAVLOV, I. P.: *Twenty Years Experience in the Study of Higher Nervous Activity (Behaviour) of Animals,* Collection of Articles. Leningrad: 1938.

[3] POLETAYEV, I. A.: Signal, Sovetskoe Radio, 1958.

[4] BUSH, R., F. MOSTELLER: *Statistical Theory of Learning.* The Mathematical Theory.

[5] SHANNON, C. E. and W. WEAVER: *The Mathematical Theory of Communication.* Urbana: 1949.

[6] WALTER, V. G.: *Imitation of Life.* Scient. Amer. 1950, Vol. 182, No. 5.

[7] BERILY, E. S.: *A Seeing Electronic Animal.* Radio Electronics, 1951, Vol 23, No. 3.

[8] EICHLER, E.: *Artificial Turtle.* Radio Technik (Radio-amateur), 1955 Vol. 31, No. 516.

[9] ROSS ASHBY, U.: *Diagram of an amplifier of thinking ability.* Collection of Articles, edited by N. E. Shannon and J. McCarthy.

[10] ATTLEE, O. M.: *Conditional Probability and Conditional Reflex Machines*: Ibid.

[11] WIENER, N.: *Cybernetics.* New York: 1948.

[12] VORONIN, L. G.: *Analysis and Synthesis of Complex Stimuli in Higher Animals.* Moscow: 1953.

[13] BRAINES, S. P., and A.V. NAPALKOV: *Directional Irradiation of the Excitation Process According to Systems of Previously Developed Ties.* 18th Conference on Problems of Higher Nervous Activity, Abstracts of Reports. Leningrad: 1958.

Digitized by Google

Q | Experiments in machine learning and thinking

By T. Kilburn, R. L. Grimsdale and F. H. Sumner, Electrical Engineering Laboratories, University of Manchester (UK)

This paper describes experiments using the Manchester University computers to demonstrate machine learning and thinking. A digital computer has been successfully programmed to generate its own programmes which must satisfy certain given criteria. For these generated programmes to be novel and interesting it is essential that there be some degree of randomness in their construction.

A number of methods are available by which the machine will improve its ability in programme generation. In one of these a feedback system is employed in which the probabilities of selection of the various instructions are varied according to the success of the generated programmes, thus enabling the machine to learn the most suitable instructions to use. The machine can also learn by experience, because all successful programmes are remembered and the machine can use these to generate new programmes. In this way as the machine gains in experience, the rate of production of programmes, and the complexity of these programmes, increases.

The first criterion used was that the programmes should represent convergent series. Many programmes were produced, some of a very complex form. None of these could have been predicted and all were originally unknown to the machine. In a second series of experiments, the criteria were made less general, and the programmes had to generate a specified sequence of numbers, the first three terms of each sequence being supplied to the machine. This experiment clearly demonstrated how the machine could learn to solve more difficult problems by referring to the solutions obtained for simpler related ones.

The paper describes work now in progress to extend the above scheme in which the machine classifies the programmes it has produced and proceeds to develop its own criteria.

Expérience sur les possibilités d'une machine pour apprendre et penser. Cette communication décrit des expériences utilisant les machines à calculer de l'Université de Manchester entreprises dans le but de démontrer la manière dont une machine peut apprendre et penser.

On est parvenu à écrire, pour une machine arithmétique, un programme lui permettant se générer elle-même ses propres programmes. Ceux-ci doivent satisfaire à certains critères préétablis. Pour que les programmes résultants soient originaux et intéressants, il est absolument nécessaire qu'il y ait, dans leur construction, quelque élément aléatoire. On dispose d'un certain nombre de méthodes selon lesquelles la machine se perfectionnerait dans la génération des programmes. Une de ces méthodes consiste à employer un système de rétroaction dans lequel les probabilités de choix des diverses instructions sont modifiées selon le succès du programme généré, permettant ainsi à la machine d'apprendre quelles instructions sont les plus appropriées. La machine peut aussi apprendre par expérience, parce qu'elle tient compte de tous les programmes qui ont réussi, et peut employer ceux-ci pour produire de nouveaux programmes. Ainsi,à mesure que la machine gagne en expérience, elle produit un plus grand nombre de programmes, et de plus complexes.

Selon le premier critère dont on se soit servi les programmes devaient représenter des séries convergentes. De nombreux programmes ont été produits, dont quelques-uns de forme très complexe. Aucun n'aurait pu être prévu, et tous étaient, à l'origine, inconnus à la machine. Dans une deuxième série d'expériences, on a posé des critères moins généraux, et les programmes devaient générer une série déterminée de nombres, la machine ayant reçu les trois premiers termes de chaque série. Cette expérience montre comment la machine pouvait apprendre à résoudre des problèmes plus difficiles en se référant aux solutions obtenues dans le cas de problèmes apparentés, mais plus simples.

Cette communication présente un travail en cours d'exécution, destiné à développer le projet décrit ci-dessus, dans lequel la machine classerait les programmes qu'elle produirait et procéderait au développement de ses propres critères.

Experimente zum Lernen und Denken von Maschinen. Diese Arbeit beschreibt Experimente mit den Rechenautomaten der Universität Manchester, um das Lernen und Denken von Maschinen zu veranschaulichen.

Ein Ziffernrechner wurde mit Erfolg dahin gebracht, seine eigenen Programme zu erzeugen, die gewissen gegebenen Kriterien genügen müssen. Damit die erzeugten Programme neuartig und interessant sind, ist es wesentlich, daß bei ihrer Konstruktion ein gewisser Grad von Zufälligkeit vorliegt.

Es gibt eine Anzahl von Methoden, mit denen die Maschine ihre Fähigkeit, Programme zu erzeugen, verbessern kann. Bei einer von diesen wird ein Rückführungssystem angewandt, bei dem die Wahrscheinlichkeiten der Auswahl der verschiedenen Befehle gemäß dem Erfolg der erzeugten Programme abgewandelt werden. Das befähigt die Maschine zu lernen, die geeignetsten Befehle zu benutzen. Die Maschine kann auch durch Erfahrung lernen, indem sie alle erfolgreichen Programme speichert und durch deren Abwandlung oder Erweiterung neue Programme erzeugt. In dem Maß, wie die Maschine Erfahrung sammelt, wachsen Erzeugung und Vielgestaltigkeit der Programme.

Das erste benutzte Kriterium war, daß die Programme konvergente Reihen darstellen sollten. Es wurden viele Programme erzeugt, einige von sehr komplexer Form. Keines hätte vorhergesagt werden können, und alle waren der Maschine ursprünglich unbekannt. In einer zweiten Serie von Experimenten wurden die Kriterien weniger allgemein gestaltet, und zwar sollten die Programme eine bestimmte Zahlenfolge erzeugen, wobei die ersten drei Glieder jeder Folge der Maschine vorgegeben waren. Dieses Experiment zeigt deutlich, wie die Maschine lernen kann, schwierigere Probleme zu lösen, indem sie auf die Lösungen für verwandte einfachere Probleme zurückgreift.
Die Arbeit beschreibt Experimente, die das obige Schema erweitern. Die Maschine klassifiziert jetzt die von ihr erzeugten Programme und entwickelt ihre eigenen Kriterien.

Эксперименты над тем, как мыслят и учатся машины. В этом докладе описываются эксперименты, проведенные с вычислительными машинами Манчестерского университета с целью показать, как учатся и мыслят такие машины.
Удалось организовать таким образом цифровую вычислительную машину, чтобы она вырабатывала свои собственные программы. Машина снабжена критерием или рядом критериев, которым должны удовлетворять создаваемые машиной программы.
Для того, чтобы эти программы были новыми и интересными существенно, чтобы в их конструкции был какой-то элемент случайности. Существует ряд методов, при помощи которых машина улучшает свою способность вырабатывать программы. В одном из методов используется система обратной связи, в которой вероятности выбора различных команд меняются в зависимости от успеха вырабатываемых программ, что дает возможность машине узнать наиболее подходящие команды. Машина может также учитывать свой опыт, так как она запоминает все удачные программы и может вырабатывать новые программы. Таким образом, по мере того, как машина приобретает опыт, темп выпуска программ возрастает и сложность программ увеличивается. Первый из используемых критериев состоял в том, чтобы программы представляли сходящиеся ряды. Было создано много программ, в том числе и очень сложных. Ни одной из них нельзя было предсказать и все они первоначально не были известны машине. Во второй серии опытов критерии были сделаны менее общими; программы должны были вырабатывать заданную последовательность чисел, причем три первых члена каждого ряда вводились в машину.
Этот эксперимент ясно показал каким образом машина может научиться решать более сложные задачи, обращаясь к решениям, полученным для более простых задач.
В докладе описана также проводимая в настоящее время работа, направленная на расширение описанной выше схемы — машина классифицирует выданные ею программы и затем разрабатывает свои собственные критерии.

Experimentos para simular el aprendizaje y el pensamiento por una máquina. Este trabajo describe unos experimentos hechos con las calculadoras de la Universidad de Manchester para simular el aprendizaje y el pensamiento de una máquina.
Una calculadora digital ha sido programada con éxito de modo que ha logrado generar sus propios programas, de acuerdo con ciertos criterios dados. Para que estos programas resultados sean nuevos e interesantes es necesario que haya algun elemento aleatório en su construcción.
La máquina dispone de varios métodos para perfeccionar su generación de programas. Se puede emplear un sistema de realimentación en el cual las posibilidades de selección de los instrucciones cambian según el éxito de los programas generados, de modo que la máquina aprende cuales son las instrucciones más utiles. Por otro lado la máquina puede aprender experimentalmente ya que todos los programas satisfactorios son recordados y ella dispone de estes para lograr nuevos programas. De este modo la velocidad de producción de programas así como la complejidad de los mismos aumentan.
El primer criterio exigió que el programa representara una serie convergente. La máquina producia muchas de tales series, algunas de las cuales eran muy complicadas. Ningunas podrian estar predichas y todas eran desconocidas por la máquina. En una segunda serie de experimentos se han hecho más específicas las éxigencias para un buen programa. Aplicaciones sucesivas del programa tenian que producir los términos de una serie algébrica cuyos tres primeros términos eran dados. Este experimento pone de manifiesto que la máquina puede aprender a resolver las problemas más dificiles mediante la referencia a los resultados ya obtenidos con problemas más fáciles.
Se describe el trabajo para lograr la clasificación por la máquina de los programas generados a fin de desarrollar sus proprios criterios.

1. Introduction

The work described in this paper was started in February 1957, and some of the results were obtained on the Manchester Mark I Computer in that month. Further work was delayed until the arrival of the Mercury Computer.
A digital computer is a high-speed calculating machine which must be supplied with a *programme* or list of instructions before it can operate. These programmes are normally written by the human user of the computer. The programme contains all the ideas and thoughts of the user pertaining to the problem to be solved and the computer acts precisely according to his wishes. It is interesting to note that, when a computer has been equipped with a programme for, inverting matrices, it becomes a machine for doing this particular mathematical operation. Thus a programmed computer may be regarded as a machine. This paper is concerned with a system by which a programmed computer—the *master machine* can construct programmes which can be performed on a *sub-computer* within the main computer. These constructed programmes are required to have certain characteristics. A programmed digital computer normally produces a set of numerical results, but the master machine produces new programmes. In view of this, the master machine is considered to be capable of *thinking*, since the production of these new programmes would have required some creative thought by a human being. The master machine also demonstrates *learning* ability because it is able to make improvements on its attempts at constructing new programmes as a result of experience. It is realised that the definitions of learning and thinking are subjects of controversy and the above claims will not satisfy all critics but it is hoped that the terms will be justified by the production of new and useful ideas by the master machine. The results so far obtained are of an elementary nature, but indicate that the system is powerful and has considerable possibilities.

2. Basic concepts

The master machine operates to generate programmes which satisfy a criterion. In the work described, the criterion has been supplied, but in future it is intended that the master machine will be able to produce its own criteria. The criterion defines the problem or class of problem for which the constructed programmes will be solutions.
The most important aspect of the work is the design of the master machine which must be efficient in producing new constructed programmes. It is difficult to measure the

efficiency of such a system; for example, it may not be desirable to produce a large number of simple programmes in a short time, but preferably to develop a few complex and interesting programmes. One of the essential features of the programme generating scheme of the master machine is a certain degree of randomness, for without this no novel work could be done. At the same time, a purely random system is uneconomical and uninteresting. For progress to be made the master machine must be able to benefit by experience and it must have some form of learning ability. Further improvement to the programme construction system is obtained by an introspective process whereby successful constructed programmes are examined and the programme constructing system is modified to improve future programme generation.

The sub-computer on which the constructed programmes are performed has features similar to a normal digital computer and actually consists of an interpretive programme performed on the computer proper. The sub-computer has an accumulator register A, several storage registers S_r, and an arithmetic unit. The basic instruction list as first used is as follows:

1) Replace the number in the accumulator by the *sum* of the numbers in the accumulator and the r^{th} storage register $A' = A + S_r$
2) Replace the number in the accumulator by the *difference* of the numbers in the accumulator and the r^{th} storage register $A' = A - S_r$
3) Replace the number in the accumulator by the *product* of the numbers in the accumulator and the r^{th} storage register $A' = A \times S_r$
4) Replace the number in the accumulator by the *ratio* of the numbers in the accumulator and the r^{th} storage register $A' = A \div S_r$
5) Replace the number in the r^{th} storage register by a *copy* of the number in the accumulator $S'_r = A$
6) Replace the number in the accumulator by a *copy* of the number in the r^{th} storage register $A' = S_r$

The constructed programmes are produced by forming a sequence of instructions chosen from this list. Each instruction has two parts: the function which indicates which of the above six operations is required and the address (r) which specifies the storage register involved. One of the storage registers contains the cycle count n. This is set equal to one initially and is increased by one every time the constructed programme is obeyed. The other storage registers are filled with random numbers within a predetermined range. The way in which the constructed programmes are performed and tested against the criterion depends on the actual criterion in use.

3. Programmes with simple convergence

The first criterion used is called the convergence criterion. The accumulator is set equal to zero and the constructed programme is obeyed. After obeying the last instruction of the programme the number in the accumulator is taken as the first answer A_1. The programme is obeyed again, starting with the accumulator equal to A_1 and the second answer A_2 is obtained, and similarly for A_3 and A_4. The convergence criterion defines a successful programme as one for which the answers satisfy the inequality:

$$|A_4 - A_3| < |A_2 - A_1|$$

This is referred to as a convergence criterion because it is normally satisfied by programmes which give progressively smaller answers each time they are performed. This particular criterion was chosen for two reasons: firstly because it will be satisfied by a large variety of different programmes, and will give the master machine a fairly unrestricted field of operation, and secondly because convergent processes are of interest in a number of fields (for example in feedback systems where it is required to minimise the error quantity).

A purely random method of programme construction is one in which programmes of fixed length, say ten instructions are built up from instructions with both function and address parts chosen randomly. This method, whilst being capable of yielding results, is time wasting and has no special features of interest. In a preferred method, the programme is tested against the criterion as each instruction is added to the programme. A further variant is to try several different instructions in the same positions in the list. The first programmes to be produced represented series of the type

$$a(1 + b + b^2 + \cdots)$$

with various values for b all less than one. These series result from the presence of two instructions of the following types in the constructed programme:

$$A' = A + S_0$$
$$A' = A \div S_1$$

If storage register S_0 contains the number c and S_1 contains d then with the accumulator initially containing zero, the number in the accumulator at the end of the first run of the programme is $c/d = A_1$ and the value at the end of the second run is $(c/d + c)\,1/d = c/d^2 + c/d = A_2$. Provided c/d is less than one the programme clearly satisfies the criterion.

Whilst these results are simple it is important to remember that they are original so far as the master machine was concerned and whilst they could have easily been predicted, this was not in fact the case. The production of such programmes is very rapid and a wide variety of different forms are obtained. During a particular run of 20 minutes of the Mark 1 Computer[1]) of which time, 15 minutes was spent in printing the constructed programmes, the following results were obtained:

1) $A_n = \dfrac{108}{n}$

   where A_n is the content of the accumulator at the end of the n^{th} run.

2) two series of the type $A_n = a(1 + b + b^2 + \cdots b^{n-1})$

3) $A_n = \dfrac{A_{n-1}}{an} - n$

$$= \frac{1}{n!}\left[\frac{1\cdot 1!}{a^{n-1}} + \frac{2\cdot 2!}{a^{n-2}} + \frac{3\cdot 3!}{a^{n-3}} + \cdots\cdots n\cdot n!\right]$$

A number of unsuccessful programmes were also obtained. These were programmes which reached the, then maximum length of eight instructions without satisfying the criterion. The results produced in this manner were interesting and some of them novel. However, much complex mathematics is derived by extending simple concepts. Accordingly, the master machine is arranged to continue building up programmes after these have satisfied the convergence criterion. As an example of this, a programme of four instructions satisfied the convergence criterion giving the result

$$A_n = A_{n-1} \times \frac{100}{174} - 213 - 100(n + 1)$$

this was subsequently transformed by the addition of four instructions into a new programme, giving the result

$$A_n = \frac{A_n - 1}{174} - 2\cdot 13 - (n + 1) - \frac{199}{(n + 1)}.$$

It is often the case that certain instructions in the programmes are irrelevant, that is they do not contribute to

[1]) The Manchester University Mark 1 Computer had an addition time of 1.2 msec for two forty digit binary numbers.

Original from

the final result, e.g. when the contents of the accumulator is copied into a register which is not subsequently used. These irrelevant instructions are removed from the programme by the master machine, but it is undesirable to do this until the programme build-up is completed. In one particular case the result

$$A_n = \frac{226}{n}$$

was obtained with a programme of 11 instructions of which 9 were irrelevant. However, the addition of two further instructions transformed the programme to give a result

$$A_n = \frac{226}{(n+1)(a A_{n-1} + b)} \quad a = \frac{138}{129} \quad b = \frac{226}{129}$$

making use of seven of the previously irrelevant instructions. It is interesting to observe that this production of irrevelant work also occurs in human thought processes and indeed some of the ideas which are, at first, apparently irrelevant prove to be of value later.

4. Introspection

In all the above examples the choice of the instructions was made at random. To improve the system further, a form of introspection is now introduced. This is arranged so that the master machine chooses the instructions in a way which depends on previous successful choices. The instructions have two parts, functions and addresses, each of which may be selected on a probability basis. Functions or addresses which have resulted in successful programmes in the past may be given a greater probability of being subsequently chosen. To illustrate the effect of choice of functions on a probability basis, all other variable factors are removed and programmes of fixed length (10 instructions) are constructed. The probabilities are varied automatically by the master machine. Each of the six functions has a number p associated with it and the p's may have any positive value provided the sum of all the p's is constant. Initially all the p values are equal. The relative values of p give the probabilities of the functions being chosen. With a given set of probability values the master machine constructs 300 programmes and tests each one against the convergence criterion noting how many are successful. At the end of the run, the master machine makes a change in the relative probabilities of the functions and a further run of 300 programmes is performed. If the change in probability distribution results in an increase in the number of successful programmes, a similar change is made again, but if the number of successful programmes decreases, the system reverts to the previous state and the effect of a different change is investigated. The results are shown in fig. 1.

The first graph shows the number of successful programmes (out of a maximum possible of 300) for each run. Every run has a different set of probabilities for the functions, and these are shown in the second graph. p_i represents the probability of selection of function 1, etc. These results show that after sixteen runs, the number of successful programmes is twice as many as when the functions have equal probability. Further changes in probability distribution cause the number of successes to oscillate about a value of 110 successful programmes. This set of experiments demonstrates the ability of the master machine to learn by experience. Thus, by using a type of feedback mechanism, the system is altered to increase the proportion of successful programmes constructed. It can be seen that the probability of the division function has increased, the probability of the operation which replaces the contents of the accumulator by the contents of a given store line has been reduced and the probabilities of the remaining functions are almost unchanged. The increase in the probability of

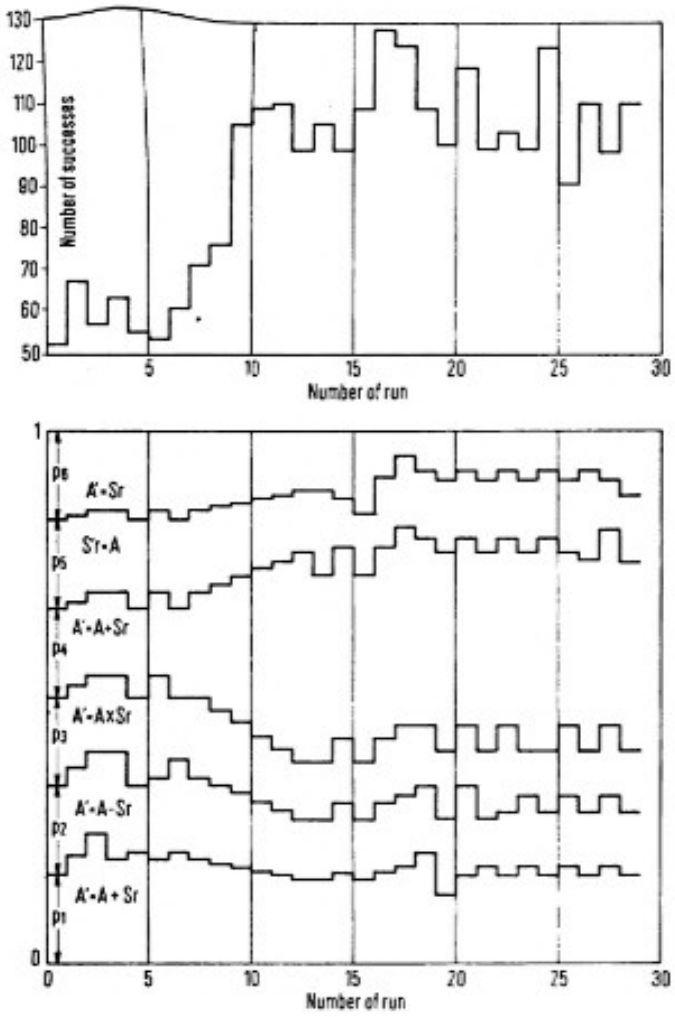

Fig. 1.

the division function is an obvious result. The reason for the decrease in the other function is that the result of prior instructions which might have led to a convergence has been overwritten by refilling the accumulator from a store line. This particular set of experiments is of an elementary nature, but clearly demonstrates the operation of a machine which can improve itself. The same principle could be extended to include the choice of the address and, further, to make improvements by observing the structure of previous successful programmes, noting such features as sequences of instructions.

5. Looped programmes

The master machine is in no way restricted to the set of instructions already given. Experienced programmers will be familiar with the technique of writing a set of instructions which are repeated many times. These 'loops' of instructions are widely used in man-made programmes, and the master machine was extended to include their use. The loop function is added to the six existing instructions. The number of times the loop is to be repeated is specified by the number in the storage register referred to in the instruction, and the number of instructions in the loop is chosen on a probability basis. To illustrate the use of the loop instruction a constructed programme in which it was used will be described.

The first success was obtained with a programme of three instructions

$\rightarrow A' = A + S_n$	where S_n is the storage register containing the cycle count.
repeat once	
$A' = A \div S_n$	

This gave the result

$$A_n = \frac{A_{n-1}}{n} + 2$$

The master machine then added further instructions to the programme, giving:

→ $A' = A + S_n$
repeat once
→ $A' = A \div S_n$
$A' = A \div S_n$
$S_1' = A$
$S_2' = A$
$A' = S_3$ (S_3 contained zero)
$A' = A + S_n$
$A' = A \times S_1$
repeat n times

The new programme gives

$$A_n = \frac{A_{n-1}}{n^n} + \frac{2}{n^{n-1}}$$

It will be seen that this programme, in common with most others, contains irrelevant instructions. When the construction of the programme is completed, the master machine removes all irrelevant instructions and prints the programme together with an algebraic representation of the action of the programme.

The programmes so far described are short, but nevertheless show interesting examples of convergence. As the programmes are allowed to grow they become very complex indeed. As an example, one such programme led to the result

$$A_n = \left[\left[\left[\left[(B_{n-1} \cdot n - c)\frac{1}{n^3 d} + \frac{A_{n-1}}{n}\right] - n - f\right]\frac{1}{A_{n-1}} + 2\right]\frac{1}{n^2} - b - 2n\right]\frac{1}{e}$$

$$\text{where } B_n = \frac{A_{n-1} \cdot (n - c)}{n^3 d} + \frac{A_{n-1}}{n}$$

This had 29 instructions, of which 6 were irrelevant.

The master machine has demonstrated that in satisfying the convergence criterion it can generate a large number of new and hitherto unknown ideas concerning convergence. For the programmes to be of value it is necessary for the master machine to classify them. The results obtained, however, are difficult even for a human to classify, and a discussion of classification systems will be left until the end of the paper.

Original from

6. Sequences

One feature of the convergence criterion is that the master machine can satisfy it easily. This is in some ways desirable, because it means that the master machine is fairly unrestricted in its attempts to produce programmes. However, the master machine is also able to produce programmes satisfying more restricted criteria. This aspect of the work, whilst similar to that described previously, does demonstrate certain abilities of the master machine more clearly.

In the restricted criteria a set of numbers or letters are given which are the first few terms of a sequence. The master machine now generates programmes which produce this number or letter sequence as successive results, and by repeating the programme, further terms of the sequence will be generated.

The programme generation system used differs in some ways from that described previously. There is a set of input registers, I_1 to I_n in which the first terms of the defining sequence (the criterion) are stored, also a set of output registers O_1 to O_n which must be made equal to I_1 to I_n by the programme. Certain of the storage registers are used to hold information supplied to the machine, and cannot be altered by the programme. The facility of B-modification of the instructions is also included; only one B-register is used, and if an instruction is to be modified then the content of the B-register is added to the address part of the instruction before it is obeyed. The programme starts with the accumulator, B-register and O-registers equal to zero. Four additional functions are also available

7) Add the contents of S_r to the B-register $B' = B + S$
8) Copy the contents of A to the next available O-register $O' = A$
9) Set the contents of the accumulator equal to zero $A' =$ zero
10) Repeat the programme R

Function 7 can occur at any point in the programme and is selected in the same way as the previous six functions. Instructions 8, 9 and 10, which do not require addresses, are only used at certain points in the programme.

A programme is generated in the normal way by forming a sequence of instructions. After the addition of each instruction, the programme is obeyed starting with the accumulator containing zero. At the end of the programme, the contents of the accumulator are compared with the contents of the first I-register. If they differ, programme generation is continued. However, if they are the same, an 8 instruction is added to the programme and programme generation may be continued. Alternatively, the programme can be terminated by any of the following sequences of instructions:

10; 9, 10; 7, 10 or 7, 9, 10.

In the case of termination, the programme is obeyed repeatedly until all the necessary O-registers have been correctly set. In the event of a disagreement the programme is partially rewritten starting at a suitable point. When the programme has satisfied the criterion, both programme and criterion are remembered by the master machine and a coded form of the programme is printed out.

The type of sequences presented to the master machine are exemplified by the following:

A B C C B A A B C C .....
A A B B C C .....
A 1 B 2 C 3 .....
1 2 3 4 .....
1 3 5 7 .....
1 3 6 10 15 21 28 .....
1 4 9 16 .....
2 6 12 20 .....
2 24 108 320 .....

For the first three sequences the machine was supplied with the first few letters of the alphabet; these were written by the machine into certain of the storage registers.

The numerical sequences are of special interest because in generating these the value of past experience, and the learning ability of the master machine, are demonstrated. The fourth sequence and the last three sequences are similar because they involve power series in N. They represent, respectively, n, n^2, $n^2 + n$, and $n^3 + n^4$. The master machine was arranged, in the first instance, to generate programmes to produce these sequences without recourse to any past experience.

The programme for the first sequence was produced in five seconds, but programmes for the remainder failed to appear within five minutes. However, when the machine was allowed to use past experience the results shown in fig. 2 were obtained.

time (secs)	
0	Sequence 1, 2, 3 given.
5	Answer 4 produced.
18	Sequence 1, 4, 9 given.
20	Answer 16 produced.
29	Sequence 2, 6, 12 given.
30	Answer 20 produced.
38	Sequence 2, 24, 108 given.
41	Answer 320 produced.

Fig. 2.

In the intervals between the master machine producing answers and being given new sequences, it develops and produces new non-trivial programmes by adding to and modifying programmes already produced. In other words, the master machine is allowed to *think* about questions it has been asked.
This learning ability was further demonstrated in another experiment in which two sets of eight sequences were used. The first (set A) was fairly simple, whilst the second (set B) was more difficult.

Set A	Set B
1, 2, 3 ..... (n)	1, 3, 7 ..... $(n^2 - n + 1)$
1, 4, 9 ..... (n^2)	1, 6, 15 .... $(2n^2 - n)$
2, 6, 12 ..... $(n^2 + n)$	3, 5, 7 ..... $(2n + 1)$
2, 5, 10 ..... $(n^2 + 1)$	3, 16, 45 ... $(n^3 + 2n^2)$
0, 2, 6 ..... $(n^2 - n)$	0, 4, 18 .... $(n^3 - n^2)$
0, 3, 8 ..... $(n^2 - 1)$	1, 16, 81 ... (n^4)
1, 8, 27 ..... (n^3)	2, 3, 4 ..... $(n + 1)$
2, 12, 36 .... $(n^3 + n^2)$	0, 8, 54 .... $(n^4 - n^3)$

Fig. 3.

In the first case the master machine had no past experience and was only permitted to make use of the programmes generated to represent other sequences in the same set. Under these conditions the machine produced programmes for all members of Set A in 1 min. 15 secs., but failed to complete the eight programmes for Set B in 5 mins. In the second instance the master machine was able to make use of past experience, as in the first experiment. That is, it was able to make use of any programme it had produced to generate given sequences, and in addition to be able to use any one of a number of programmes the master machine had generated by forming programmes of a type similar to the successful programmes.
The experiment was then repeated, the master machine being given Set A first, which it again produced in about 1 min. 15 secs. Set B was then supplied, and the master machine now produced programmes which generated all these sequences in under ½ min. In producing the programmes for Set B, the master machine made use of the 8 successful programmes for Set A, some 39 programmes developed as variations on this set, and a further 67 programmes which were developed as variations on the programmes of Set B as they were produced. All these sequences were polynomials of n and the probabilities of the addition, subtraction and multiplication functions, and the store register containing n increased rapidly. When sequences of a different type were given following Set A and Set B the master machine was slow in generating programmes until a new probability distribution for the functions and addresses had been established.

7. Problem solving

The experiments described in this paper show that it is possible to design a machine with a certain degree of thinking and learning ability. The master machine has produced a large number of convergent series which were hitherto unknown to the machine, or indeed to the designers. The rate of production of the programmes representing these series is very high, and the range of different types is considerable. The operation of generating a programme to satisfy a criterion is in some ways akin to one aspect of the human processes of invention. A human inventor when faced with a problem, sifts and arranges the information in his memory to produce ideas, and applies tests to see if any ideas will solve the problem. It appears possible that a thinking machine of the type described in the paper could be constructed, which would be able to solve problems in a particular field such as electronics or nuclear engineering, provided such a machine was supplied with sufficient information about the subject, together with any other basic information required.
The demonstration that a machine can learn to solve problems is also an important feature of this work. Learning can be of two types; learning by experience and learning by teaching. The master machine learns by experience when it makes use of programmes it has developed in the past. The second form of learning is demonstrated when the machine is assisted in solving difficult problems by being given easier problems first.
The master machine has certain *built-in* abilities. It is able to generate programmes consisting of certain computer-type instructions, and a sub-computer is available on which these programmes may be performed. The sub-computer has facilities such as addition, subtraction, multiplication etc. The master machine could have been given fewer facilities to start with, but then it would have only been capable of performing very simple operations. Indeed it would be preferable for the master machine to have more built-in facilities, because the present number is very small when compared with the human brain.
The experiments decribed in this paper are of an elementary nature, and represent a start in the construction of a thinking machine. The system is now being improved and extended, and some of the ideas in this direction are now described.

8. Classification

Many of the constructed programmes will bear a relation to others, and some form of classification system is required. There are a variety of different ways in which this may be done. The classification may be based on the constructed programmes themselves. The type of instructions and the order in which they are arranged may be noted, and programmes having common features may be grouped together. This amounts to a pattern analysis of the programmes. The ability to recognise patterns of instructions within programmes and to make use of these patterns in future programmes will give the machine further creative ability. An algebraic representation is already obtained for programmes now produced by the master machine during the process of removing redundant instructions; this representation will be particularly suitable for pattern analysis.
The classification may also be based on the criteria. The master machine can observe characteristics of the criterion, for example, it may require an inequality to be satisfied or a sequence to be continued. Thus when the machine has a similar criterion on a future occasion it will be able to recall previous solutions and methods of solution used for similar problems. In this connection it will be very desirable for the machine to have a limited understanding of a language with which it may communicate with human beings. In this way names can be given to the various classes of programmes which can subsequently be referred to by humans. The language may also be used to give advice to the machine, as it is realised that the machine will require a considerable amount of teaching to be able to solve significant problems.

The machine may also be left to work by itself, and the master machine is at present being arranged to work in this manner. In the experiments described the machine produces programmes to satisfy given criteria, whereas in the system the criteria themselves will be produced by the master machine. The criteria will be based on certain mathematical, logical, and other concepts and relations. These would include such ideas as *greater than, equal to, similar, if, but, and, or, then*, etc., and concepts such as *order* or *sequence*. The master machine will manipulate these items to form criteria and proceed to attempt to generate programmes to satisfy criteria.
A master machine which invents programmes to satisfy criteria produced by itself would indeed be a thinking machine on almost any definition. Purposeful thinking to human advantage can only follow if the machine is given contact with the outside world, either directly or with the assistance of human beings.

9. Discussion

G. R. Boulanger (Belgique): Je voudrais féliciter MM. Kilburn, Grimsdale et Sumner d'avoir osé utiliser, dans ce Congrès, l'expression «thinking machine» (machine à penser). C'est une audace. On remarquera, en effet, la contradiction qui existe entre les possiblités que laissent entrevoir MM. Kilburn, Grimsdale et Sumner, et ce qui a été dit au cours de la séance inaugurale du Congrès. Il y a été affirmé en effet qu'une machine ne peut travailler que suivant des programmes établis par l'homme, que tout accès au plan de la création intellectuelle lui est interdit etc. De telles affirmations sont dangereuses, car leur caractère est purement gratuit. Que les machines à programmes travaillent de cette manière, nul n'en doute. Mais il est d'autres types de machines. Et les prises de position que j'incrimine sont d'autant plus regrettables qu'elles sont affichées par des personnalités plus éminentes et dans des circonstances plus officielles. Car elles tendent à jeter le discrédit sur les efforts des chercheurs qui ont précisément choisi pour hypothèse de travail la non-limitation à priori des possibilités de la machine.
On dira qu'il s'agit dans les deux cas d'hypothèses et que l'on ne peut apporter le preuves pour aucune. C'est exact, mais il y a une différence: la première attitude (celle de la négation) est stérilisante tandis que ceux qui ont adopté la seconde nous apportent chaque jour de nouvelles raisons de croire à son bien-fondé.
Il n'est pas opportun d'amorcer ici un débat sur les possibilités des machines mais je crois que, pour éviter des malentendus ultérieurs, ces choses devaient être dites au cours du présent Congrès.

R | A machine model of recall

By Mary E. Stevens, National Bureau of Standards, Washington D.C. (USA)

A machine model of certain logical recall operations, involving both pattern recognition and a limited degree of machine learning is described. The model consists of an initial vocabulary of terms (nouns, adjectives, and proper names) and stored records of certain of their semantic and logical interrelationships, together with routines for various operations upon the machine's store of "knowledge" as available at any given time. In the operation "Define," the machine defines a given input term with respect to other terms that are applicable to it. In "Extend," the machine lists specific examples of a given generic term. In "Locate," the machine searches for any term in its vocabulary related to a given input term in the sense of identifying the geographic location, if any appropriate to that term. In "Match," several input terms are compared to find common reference to other terms and the vocabulary is then searched for any additional term that thus matches the input terms.
These and other operations are used to illustrate potentialities for new machine aids to information retrieval, literature search, etc. Other operations provide for a measure of "learning" of new terms as well as for "forgetting" of other terms. New terms are accepted either by tentative assignment of relationship references, or by routines calling for man-machine intercommunication during the operation and for operator feedback, including "rewards" for correct answers.
Experimental results obtained with SEAC are reported, and the implications of the tests of the model for information retrieval, intelligence testing, etc., are discussed.

Modèle de rappel réalisé sur machine. L'auteur décrit un modèle de certaines opérations de rappel logique d'informations enregistrées, exigeant de la machine à la fois qu'elle puisse identifier des structures et, dans une certaine mesure, apprendre. Ce modèle comprend un vocabulaire initial (noms, adjectifs et noms propres) et un répertoire enregistré de certaines relations sémentiques et logiques entre les termes de ce vocabulaire, ainsi que les programmes de diverses opérations à effectuer sur le stock de «connaissances» disponibles à un moment donné. Lorsqu'elle effectue l'opération «définir», la machine définit le terme qui lui est proposé par rapport à d'autres termes apparentés. Si on lui commande de «développer» la machine énumère des exemples précis du terme générique donné. Lorsqu'elle reçoit l'ordre de «localiser» la machine cherche dans son vocabulaire, parmi tous les termes se rapportant au terme proposé, celui qui en indique la situation géographique. Si elle reçoit l'ordre d'«apparier» elle compare les différents termes qui lui sont proposés pour y découvrir une référence commune à d'autres termes éventuels, puis dépouille son vocabulaire pour en extraire tout terme additionnel qui puisse de la même manière être rattaché aux mots d'entrée.
En décrivant les diverses opérations dont cette machine est capable, l'auteur montre ce qu'on peut attendre de nouveaux auxiliaires mécaniques en matière de rappel d'informations enregistrées, de recherches documentaires, etc.
D'autres opérations permettent dans une certaine mesure l'«apprentissage» de termes nouveaux ainsi que l'«oubli» d'autres termes. La machine enregistre les termes nouveaux soit en leur affectant provisoirement les références, soit à l'aide de programmes exigeant des échanges d'information homme — machine au cours de l'opération ou une réponse (feed-back) de l'observateur qui «récompensera» notamment la machine si elle répond bien.
L'auteur rend compte des résultats d'expériences faites avec la machine SEAC et examine les conséquences à tirer de différents essais auxquels le modèle a été soumis (rappel d'informations et tests d'intelligence notamment).

Ein Maschinenmodell zur Beantwortung von Fragen. Es wird ein Modell für bestimmte logische Operationen zum Wiederauffinden von gespeicherten Informationen beschrieben, das auch Zeichen erkennen und bis zu einem gewissen Grad lernen kann. Das Modell besteht aus einem anfänglichen Vokabular von Ausdrücken (Substantive, Adjektive und Eigennamen), aus gespeicherten Aufzeichnungen eines bestimmten Teiles ihrer semantischen und logischen Beziehungen und aus Programmen

Digitized by Google
Original from