Psychological Review
1973, Vol. 80, No. 2, 126–138

A THEORY FOR THE INDUCTION OF MATHEMATICAL FUNCTIONS[1]

L. ROWELL HUESMANN[2] AND CHAO-MING CHENG
Yale University

A detailed theory is presented for how humans induce mathematical functions to explain observed data. The theory is based on both solution times and verbal protocols. The theory proposes that induction is a heuristically directed generate-and-test process. The order in which hypotheses are generated is mostly independent of the data. Each hypothesis is maintained until it is negated, but a false hypothesis that matches part of the data is frequently retested. A computer simulation program, incorporating these processes, accurately predicted solution times for subjects on different sets of problems without any changes in parameters. In addition, the program's solution protocols were indistinguishable from human protocols.

An individual engaged in solving a difficult problem is faced with two tasks: the surface task is simply to solve the problem, but the deeper coincident task is to derive general heuristics, algorithms, and methods of representation that can be applied to new problems. Obviously, this process of extracting information out of specific examples to form general rules is induction. On the surface, induction in this situation seems quite different from most concept-learning situations constructed in the laboratory. The problem solver knows only that algorithms, heuristics, and representations exist which should be employed when indicated by certain ill-defined quantitative features from a large set of features related to the problem. His goal is to learn which features are relevant, and what the functions are that map the features into the methods.

Most strategies that a problem solver derives may be stated in many different equivalent forms. One such form would be a mathematical function that maps observed values of relevant stimulus variables into observed utilities of solution methods. Representing strategies as mathematical functions, however, is particularly useful if one follows the conception of Newell and Simon (1961) that problem solvers form tables of connections from characteristics of the task environment to solution methods.

Our view is that a problem solver begins the learning process by constructing part of a table of connections between states of the task environment and solution methods. When enough data have been collected in such a table, he induces rules that specify which solution methods are good in each situation. The induction of these rules based upon the data in the table of connections could be accomplished by an induction process that maps classes of stimulus values into utilities for solution methods. More formally, we can say that to induce a rule for solving a problem one must find a function $f(x_1, x_2, \ldots x_p)$, such that if $x_1, x_2, \ldots x_p$ are stimulus variables and y is a variable representing the utility of a particular solution method, then $f(x_1, x_2, \ldots x_p) = y$ for all observed instances of $(x_1, x_2, \ldots x_p, y)$.

By modeling the derivation of problem-solving strategies as a case of inducing a mathematical function, we have not intended to imply that humans necessarily see strategy formation in such a light. Rather, we wished to show that the induction of a mathematical function is typical of the type of induction task that problem solvers face. Thus, an understanding of how mathematical functions are derived should contribute to an understanding of how problem-solving strategies are learned.

[1] This research was supported in part by U. S. Public Health Service Research Grant MH-19331-01 from the National Institute of Mental Health.

[2] Requests for reprints should be sent to L. Rowell Huesmann, Department of Psychology, Yale University, 333 Cedar Street, New Haven, Connecticut 06510.

INDUCTION OF MATHEMATICAL FUNCTIONS AND CONCEPT FORMATION

Clearly, the problem of inducing a mathematical function is a concept formation task. Yet the induction of a general mathematical function to explain observed numerical data is different enough from the concept-learning tasks studied in the laboratory that little can be inferred with certainty. Traditional concept formation experiments have investigated the attainment of Boolean functions rather than arithmetic or other types of functions. Hunt, Marin, and Stone (1966) have suggested a general model for induction based on concept-learning studies, but the assumptions underlying that model are violated when the function being induced is not Boolean. In particular, the induction of arithmetic functions differs from learning Boolean rules for pattern classification in that (*a*) the number of output classes may be infinite in the arithmetrical task (e.g., the real numbers), (*b*) the input attributes may assume an infinite set of values in the arithmetical task, and (*c*) there may be no limit of possible arithmetical rules for mapping the input variables into the output variable.[3] However, many of the processes employed in Boolean concept learning are suggestive of processes that could be employed in deriving general mathematical functions. The hypotheses-testing theories of Restle (1962), Bower and Trabasso (1964), Gregg and Simon (1967), and their supporting experimentation suggest that induction is primarily a heuristic generate-and-test process.[4]

Studies of serial-pattern processing (Feldman, Tonge, & Kanter, 1963; Gregg, 1967; Leeuwenberg, 1969; Restle, 1970; Simon & Kotovsky, 1963; and Simon & Sumner, 1968) also suggest that subjects construct formal rules for specifying sequences by means of heuristic generate-and-test processes. For example, the Simon and Kotovsky model for series-completion tasks asserts that subjects first search for a regularity that defines a cycle and then generate and test hypotheses until one is found that explains the series. Recently, Simon (1972) has shown that these various theories proposed to explain serial pattern processing are essentially the same, and all include the generation of a formal rule by the subject and his production of new sequences with the rule.

Other evidence for this view of induction as a heuristic search process stems from artificial intelligence research. As Plotkin (1971) and others have pointed out, problems in induction are theoretically insoluble in that no *general* method can exist for finding laws to explain particular phenomena. One can only prove that a particular must follow from a generality, not vice versa. It should not be surprising, then, that many computer programs which have been built to perform induction employ heuristic generate-and-test techniques. Pivar and Finkelstein's (1964) program for inducing the serial pattern in a sequence of letters or numbers uses a variety of clever devices to generate plausible pattern descriptions for the given series. Each is a heuristic technique not guaranteed to succeed, but the combination of them constitutes an impressive program. Similarly, Williams (1972) and Buchanan, Sutherland, and Feigenbaum (1969) employed heuristic generate-and-test processes to solve induction problems typical of intelligence tests and organic chemistry, respectively. On the other hand, induction programs, particularly in the pattern recognition field, have frequently used techniques quite different from the hypotheses-generating algorithms mentioned above. Most notably, the use of linear discriminant functions and error correction training (Nilsson, 1965) has been prevalent.

Bongard (1970) has used these devices in a program called ARITHMETIC that "learns" discriminant functions for recognizing the mathematical functions relating one variable to several others. While such "perceptrons" may have the potential for being models of perceptual processes, their limited applicability to induction in problem

[3] With a Boolean function of p variables, the number of possible functions is equal to the number of different truth tables, that is $2^{(2^p)}$.

[4] By heuristic, we mean that the generation and testing of hypotheses is directed by rules (heuristics) designed to locate a solution rapidly but which allow the possibility that no solution will be found.

solving is well known (Minsky & Papert, 1969). Finally, one should recognize that the problem of inducing a mathematical function to explain observed data is quite similar to the problem of inducing a grammar to explain an observed language. Hence, it is worthwhile to note Feldman's (1972) proof that a program that generates grammars *in order of increasing complexity* can find the best grammar for a language.

Studies of problem solving have also uncovered information suggestive of how people proceed with induction problems. Wason (1968) asked subjects to supply the rule that governed what the next element of a numerical series would be. While the rule actually used was that the next number had to be greater, subjects generated a wide variety of complicated hypotheses. Interestingly, when a subject was told to generate the next instance in the series, he tended to try to generate an instance confirming his current hypothesis rather than one that might disconfirm.

Other evidence of (apparently) inefficient search behavior in hypothesis testing has been reported by Smedslund (1967). He found that humans do poorly at detecting correlations between dichotomous variables. Smedslund suggests that the overreliance of subjects on positive instances supporting incorrect hypotheses contributed to this deficit. At the same time, evidence exists that some subjects, when given the opportunity, use very systematic search procedures to induce rules rapidly. Duncan (1967) found that when subjects in an induction experiment were allowed to control the variation of the stimulus variables, those who varied them one at a time derived the principle most rapidly. To summarize, induction seems to be characterized by a heuristic generate-and-test process during which positive confirmations add to the credibility attached to an hypothesis. Subjects who use systematic search techniques seem to learn more rapidly.

ANALYZING INDUCTIVE BEHAVIORS

While the preceding evidence has been suggestive of how people might induce mathematical functions from observed data, the differences between most concept formation tasks studied and the task of inducing a mathematical function are not trivial. Hence, before proposing a theory, one should obtain more specific information about the processes that humans employ in solving such induction problems. To do this, we designed a problem-solving experiment in which our objective was to obtain verbal protocols and solution times for subjects who were solving problems of inducing arithmetic functions from data.

Experiment I

Problems. Nine problems requiring the induction of an arithmetic function were used in this experiment. Each problem consisted of six instances of the form $a \$ b = c$. The two operands on the left, a and b, could be mapped into the operand on the right, c, by some arithmetic expression constructed from the two operands; the operator's addition, subtraction, multiplication, division, and exponentiation; and integer constants between 1 and 4. No more than three operators were used in any expression. The nine problems and their solutions are shown in Table 1.

Subjects. Eighteen undergraduates were employed as subjects. Only those subjects were analyzed who solved at least two out of three problems. In order to obtain 18 valid subjects, 22 subjects had to be tested.

Procedure. The stimuli for one induction problem consisted of six simple mathematical equations of the form $a \$ b = c$, for example, $2 \$ 3 = 8$. Each of the six equations represented a positive instance of the correct concept (function). No negative instances were provided. The problem was printed on a large white sheet about 2 meters from the subject. The six instances were listed in one column beginning at the top of the page. All six instances were exposed simultaneously.

The subject was told before the experiment that the symbol "$" could represent one of the simple mathematical operators (+, −, *, /, ↑), or some combination of them, but not more than a four-level com-

TABLE 1

INDUCTION PROBLEMS USED IN EXPERIMENT 1

Number of operators in correct function	Number of alternative functions fitting at least half the instances of data and having no more operators than the correct function			Solution $f(A, B)$
	0	1–2	≥ 2	
1	$f(1, 0) = 1$ $f(-2, 2) = 4$ $f(3, 0) = 1$ $f(1, 4) = 1$ $f(-3, 2) = 9$ $f(2, 1) = 2$	$f(2, 2) = 4$ $f(1, 0) = 1$ $f(-1, 2) = 1$ $f(2, 3) = 8$ $f(-3, 2) = 9$ $f(-1, 3) = -1$	$f(2, 2) = 4$ $f(1, 0) = 1$ $f(-1, 2) = 1$ $f(2, 1) = 2$ $f(3, 1) = 3$ $f(-3, 2) = 9$	A^B
2	$f(2, 2) = 5$ $f(2, -1) = -1$ $f(2, 3) = 7$ $f(2, 1) = 3$ $f(-3, 2) = -5$ $f(3, -1) = -2$	$f(2, 1) = 3$ $f(0, -1) = 1$ $f(3, 2) = 7$ $f(2, 2) = 5$ $f(2, -1) = -1$ $f(2, 0) = 1$	$f(2, 1) = 3$ $f(1, 0) = 1$ $f(3, 2) = 7$ $f(-1, 3) = -2$ $f(1, 2) = 3$ $f(1, 4) = 5$	$AB + 1$
3	$f(3, 2) = 5$ $f(2, 1) = 2$ $f(3, 3) = 3$ $f(4, 3) = 10$ $f(2, 4) = -4$ $f(5, 5) = 15$	$f(3, 2) = 5$ $f(2, 2) = 0$ $f(1, 2) = -3$ $f(2, 1) = 2$ $f(3, 3) = 3$ $f(2, 4) = -4$	$f(3, 2) = 5$ $f(-2, 2) = 0$ $f(4, 2) = 12$ $f(1, 2) = -3$ $f(4, 4) = 8$ $f(-3, 4) = 1$	$A^2 - 2B$

bination. He was also told that the function might include a constant, for example, $3a - b^2$, but that any constant would be an integer between one and four inclusive. The subject was reminded of these requirements if he violated them. The subject was told in written instructions to say whatever came to his mind and to specify whatever instance he was looking at at any moment during the experiment.

While no time restriction was announced, the time for solving any one problem was limited to 25 minutes. If the subject failed to solve a problem, he was told the correct answer and then was asked to perform the next one. Before beginning work on the test problems, the subject performed three examples of different difficulties. Then, if he had no questions, he was given the first test problem. During the problem-solving period, his verbalizations were recorded, and his solution time was measured. Whenever the subject was silent for long periods, the experimenter asked him to speak up.

Each subject was tested on three problems. A 3 × 3 Latin square design (Table 1) was used with the nine cells representing nine problems. One side of the square represented the number of operators in the correct function, that is, one, two, or three, while the other side represented the number of alternative hypotheses that had no more operators than the correct function and that fit at least half of the instances of data.

Results. Two types of data were available for analysis: solution times and verbal protocols. The solution times (see Table 2) support the theory that subjects begin generating simple hypotheses and then systematically increase the complexity of the hypotheses they try. Solution times increased significantly with the number of operations in the correct hypothesis ($F = 7.88$, $df = 2/30$). In addition, the greater the number of alternative hypotheses fitting the majority of instances, the longer were the solution times ($F = 3.60$, $df = 2/30$). This latter result indicates that subjects had difficulty abandoning a wrong hypothesis that fit a major portion of the data.

Protocols. The protocol data, while less easily quantitized, contain more specific in-

TABLE 2

MEDIAN SOLUTION TIMES IN EXPERIMENT I

Number of operators in correct function	Number of alternative functions fitting at least half the instances of data and having no more operators than the correct function			
	0	1-2	≥2	$\bar{M}$
1	52.5	97.5	137.5	95.8
2	180.0	180.0	382.5	247.5
3	502.5	675.0	690.0	622.3
$\bar{M}$	245.0	317.5	403.3	321.9

Note. All times are in seconds.

formation. One way to test the hypothesis that functions with i operators are generated before functions with $i + 1$ operators is by extracting the order in which the hypotheses were generated from the protocols. Having found the order for the initial problem every subject solved, we computed a correlation between the number of operators in a hypothesis and its rank in the sequence of hypotheses. The correlations ranged from .30 to .95 for the 17 subjects with usable protocols. The average correlation was .65. Such correlations are imperfect and conservative measures since even a perfect ordering would not yield a correlation of unity.

Furthermore, many of the later occurrences of simple hypotheses were repetitions of earlier occurrences. Hence, it seems fair to say that every subject displayed a tendency to generate hypotheses with i operators before those with $i + 1$ operators. One can perceive some regularities in the discrepancies from this procedure. For example, the majority of the discrepancies on the six more difficult problems were characterized by the subject skipping ahead to try a few more complex functions and then backing up to see if he missed a simpler function.

The number of operators was not the only factor that could be seen to influence the order of generation. Within a set of hypotheses having the same number of operators, subjects appeared to have fairly rigid orderings based on a variety of rules: (*a*) Addition, subtraction, and multiplication were usually tried before exponentiation or division. For example, in only 2 of 51 problem protocols was a hypothesis with division attempted before at least two operators from the set (+, −, *) were tried. Similarly, in only 21 of 51 problems was exponentiation used before two of the others even though exponentiation occurred in the correct solutions of six of the nine problems. (*b*) Subjects displayed a clear left-to-right bias in generating noncommutative hypotheses. For example, while virtually every subject generated a/b, $a - b$, and a^b at some time, only two subjects generated b/a and none generated $b - a$ or b^a. (*c*) If an additive constant was appended to a function, it was usually done right after the function had been negated by an instance.

The frequencies with which particular hypotheses were used adds some other evidence in support of a primary generate-and-test process. If the same clues in the problem were used extensively in the same way by subjects to control what hypotheses were attempted next, one would expect to find many hypotheses used by a large number of subjects even when the hypotheses were quite complex. On the other hand, a less directed generate-and-test process with more random variation between subjects would yield extensive common use among subjects of the simple hypotheses which everyone generates and little agreement on the more complex hypotheses at the bottom of the generation tree. The distribution of noncorrect alternatives supports this latter view. While every subject tried each of five one-operator hypotheses, only two of the 23 three-operator and four-operator hypotheses used were tried by more than one subject.

Two facts about the testing of hypotheses on the data were also apparent from the protocols. In the majority of the cases a subject clearly began testing an hypothesis on either the first (top) instance or on the instance that negated the previous hypothesis. In addition, subjects more frequently retested an hypothesis that had been negated if that hypothesis had been supported by other instances previously. For example, the incorrect alternative solutions to the problems in the rightmost column of Table 1, each of which was supported by at least three instances of data, were repeatedly

tested an average of 1.1 times per subject. On the other hand, other alternatives were repeated only .2 times per subject.

Discussion. The analysis of the subjects' solution times and protocols has yielded evidence consonant with the results of other researchers and specific enough to be the basis for a theory of inductive processing.

The central finding was that subjects appear to generate and test hypotheses in an order that is independent of the data, but in accord with several well-specified heuristics. The most important of these was that functions with i operators were tried before functions with more than i operators. The evidence from protocols and solution times also supported the theory that subjects persevere on those wrong hypotheses for which some confirming evidence exists.

FORMAL MODEL FOR INDUCTION

We will now present a formal theory for how humans induce mathematical functions from data. This model is based upon the results of our induction experiment and previous concept-learning experiments. Of course, the validity of the theory must be confirmed by demonstrating its predictive utility. In order to specify the details of the model as precisely and unambiguously as possible, it was formulated as a computer simulation model called INDUCT-1. The program was designed so that its performance on a particular problem would vary from run to run in accord with the between-subject variations found. Its average performance should predict the average performance of the group.

The essence of the induction model is that subjects generate and test hypotheses in an order that is mostly independent of the data and partially determined by heuristics. To represent this process for problems of the type studied in Experiment I, the hypotheses that subjects had used in Experiment I were divided into pools. Within each pool the model selects any hypothesis for testing at random *without replacement.* However, the order in which the pools are generated is determined quite precisely in accord with heuristics discovered in Experiment I. Pools of hypotheses having i operators are tried (with a few exceptions to be noted below) before pools with $i + 1$ operators. Within hypotheses having the same number of operators, order is partially determined. In particular, (a) noncommutative operators are used in the order "a op b" before they are used as "b op a"; (b) additive constants are tried before any addition of one of the parameters is attempted; (c) addition, subtraction, and multiplication are tried before division and exponentiation; and (d) multiplication and exponentiation by a constant are attempted before the same operation by a parameter.

Within a pool the search procedure is to generate a hypothesis at random (without replacement) and test it on either the instance of data that negated the previous hypothesis or the first instance in the sequence of data. The hypothesis is tested successively on every instance in the sequence of data until it is negated or all instances are fit. If all instances are fit, the solution has been found; but if a negation is found, the current hypothesis is immediately rejected and a new one from the same pool is generated. When the current pool is exhausted, generation of hypotheses from the next pool begins. One should recognize that under this procedure the order in which hypotheses are generated is independent of the problem being solved. This basic search algorithm is specified formally in Figure 1a.

The program deviates from this basic search algorithm in only two ways. Some subjects were observed skipping certain pools and only trying them much later after more complex hypotheses had been found wanting. To model this phenomena, we associated a probability with each pool that approximated the observed probability that a subject would leave that pool on any generation trial and skip ahead to the next pool. At the same time, a probabilistic decision branch was added so that a subject would (after a minimal time had elapsed) go back to recheck simpler pools for untested hypotheses. The probabilities were estimated from the protocol data and then left unchanged during the simulations.

0.0 $I \leftarrow 1$; $J \leftarrow 1$;
1.0 Generate a hypothesis, H, randomly without replacement from POOL [I];
2.0 If POOL [I] is exhausted, go to 9.0;
3.0 With equal probability $J \leftarrow 1$ or J remains unchanged;
4.0 Test H on INSTANCE (J);
5.0 If test fails, go to 1.0;
6.0 If all instances have been tested, then *solution has been found*;
7.0 If $J = 6$ then $J \leftarrow 1$; otherwise $J \leftarrow J + 1$;
8.0 Go to 4.0;
9.0 $I \leftarrow I + 1$;
10.0 If $I >$ NUMBER OF POOLS, then *quit with failure*;
11.0 Go to 1.0;

FIG. 1a. The basic generate-and-test algorithm.

0.0 $I \leftarrow 1$; $J \leftarrow 1$;
1.0 Generate a hypothesis, H, randomly without replacement from POOL $[I]$;
2.0 If POOL $[I]$ is exhausted, then RECHECK any hypothesis in POOL $[I]$ that worked on at least 3 instances and go to 9.0;
2.1 If RANDOM FRAC < PROB SKIP [I], then go to 9.1;
2.2 If H has been tried before, then go to 1.0;
3.0 With equal probability $J \leftarrow 1$ or J remains unchanged;
4.0 Test H on INSTANCE (J);
5.0 If test fails, go to 1.0;
6.0 If all instances have been tested, then *solution has been found*;
7.0 If $J = 6$ then $J \leftarrow 1$; otherwise $J \leftarrow J + 1$;
8.0 Go to 4.0;
9.0 If RANDOM FRAC < 1/3, then RECHECK all past pools (that have not been rechecked) for skipped hypotheses;
9.1 $I \leftarrow I + 1$;
10.0 If $I >$ NUMBER OF POOLS then RECHECK all past pools that have not been rechecked and *quit with failure* if a solution is not found;
11.0 Go to 1.0;

FIG. 1b. The modified generate-and-test algorithm used in all simulations.

The second variation in the search algorithm was derived from our discovery that subjects fixated on incorrect hypotheses that fitted a fair portion of the data. Hence, whenever a hypothesis was discovered that worked for half or more of the instances, the simulated subjects would retest that hypothesis on all the instances before abandoning the current pool. These two variations are incorporated in the algorithm in Figure 1b.

The simplicity of this model should be apparent from Figure 1. For predicting solution times the INDUCT-1 model has four free parameters representing times for generating and testing hypotheses. Given that a hypothesis has n operators, the time for generating it once would be nT_g; the time for regenerating it, after it had been tried at least once, would be nT_r; and the time for testing it on one instance of data would be $T_{t1} + nT_{t2}$. While T_g, T_r, T_{t1}, and T_{t2} are free parameters, they represent times and were constrained to be positive and to obey the relation $T_g > T_r$. In other words, the time to generate a hypothesis initially must be greater than the time needed to regenerate it.

Evaluating the Model

The model was used to predict the average data on the nine problems used in Experiment I. During the simulation runs all the characteristics of the model remained fixed except for the four time parameters which were varied to achieve goodness of fit.

The simulation program produces two types of output that can be compared with the solutions humans generate. The program prints a protocol of its solution process, and it computes the simulated time needed to reach any point in the solution process.

In Figure 2 the median times obtained on 10 simulation runs are compared with the median times observed on each of the 9 problems for the human subjects. While the time parameters were varied systematically to minimize the squared discrepancies between the predicted and observed times, no analytical procedure could be used to determine the optimal values of the parameters because of the form of the model, and it is quite probable that the final values of the time parameters were nonoptimal.[5] The actual deviations between observed and predicted times obtained with these simulations are shown in Tables 3 and 4. These data

[5] The final values obtained for the four parameters were $T_g = 4$ seconds, $T_r = 1$ second, $T_{t1} = 2$ seconds, and $T_{t2} = 3$ seconds.

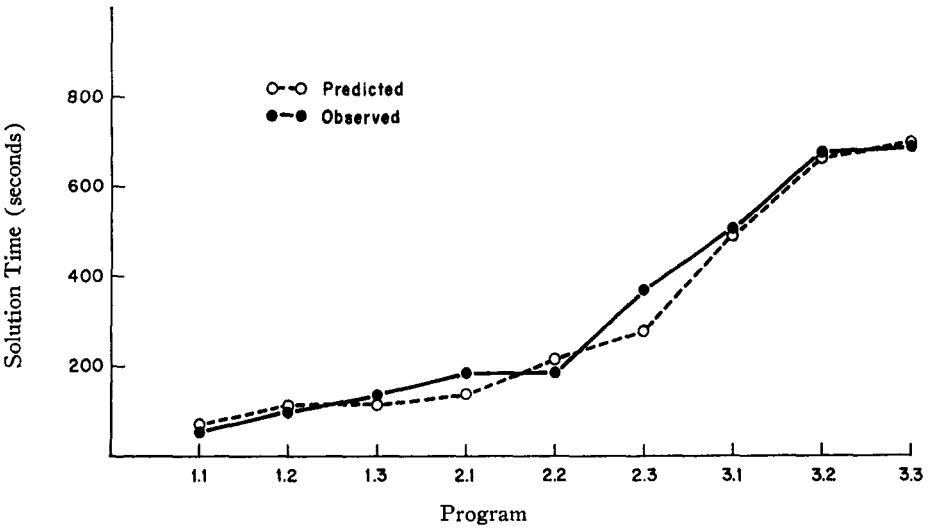

FIG. 2. A comparison of the predicted and observed solution times for the problems used in Experiment I.

seem to indicate that the model's solution times have close to the same central tendency as the observed times.

Unfortunately, most traditional quantitative measures of goodness of fit cannot be applied to this situation. One way we can measure the fit is to compute the ratio of the discrepancy between predicted and observed to the standard deviation of the observed. As can be seen from Tables 3 and 4 the discrepancy on most problems is very small compared to the standard deviation of the observed times. In addition, the distribution of predicted and observed times about the medians yields a chi-square value of only .347. Hence, it seems fair to say that the model is highly accurate in predicting the observed times.

Despite these results one may question the validity of the model on the grounds that a computer simulation model with four free parameters is not very falsifiable. While we agree that the weight to be given to the matching of fine grain statistics should be downgraded as the number of free parameters increases, other factors in a model are of equal importance in determining the fit. In this case we suggest that the constraints placed upon the simulation program by having it duplicate the processes subjects use make it highly falsifiable despite its free

TABLE 3

DISCREPANCIES BETWEEN PREDICTED AND OBSERVED MEDIAN SOLUTION TIMES

Problem	Discrepancy between predicted and observed (Δ)	Δ divided by standard deviation of observed	Observed median split
1	17.5	.290	67–33
2	9.5	.080	50–50
3	23.0	.077	16–84
4	52.0	.827	33–67
5	31.0	.204	50–50
6	107.5	.293	50–50
7	7.5	.025	50–50
8	12.0	.037	50–50
9	9.5	.031	50–50

TABLE 4

OVERALL MEDIAN TEST

Solution time	Group	
	Simulation program	Observed subjects
Below median	47	25
Above median	43	29

Note. $X^2 = .347$.

parameters. In fact, as shown below, the program duplicates the processes of humans so closely that its protocols cannot be distinguished from human protocols.

Comparing observed and predicted protocol data is somewhat difficult. While the observed and simulated protocols can be shown to be identical on many aspects, this agreement only reflects the face validity of the model which was based on the observed protocols. A sample simulation protocol is shown as follows.

The problem is:

(1) $f(2, 2) = 4$
(2) $f(1, 0) = 1$
(3) $f(-1, 2) = 1$
(4) $f(2, 3) = 8$
(5) $f(-3, 2) = 9$
(6) $f(-1, 3) = -1$

"Let's try A + B. It works on number 1. It works on number 2. It works on number 3. But it won't work on number 4. O.K., let's try another one, Ah, how about A * B. It works on number 1. But it won't work on number 2. O.K., let's try another one, Ah, how about A − B. It works on number 2. But it won't work on number 3. O.K., let's try another one, Ah, how about, . . . O.K. just scan and see what happens. A + B—It fits number 1; it fits number 2; it fits number 3; no good on number 4; no good on number 5; no good on number 6. . . . Something more difficult . . . Let's go back and check again. A + B, I've tried that. A − B, I've tried that. A * B, I've tried that. Well, I guess I didn't miss it. A^B. It works on number 1. It works on number 2. It works on number 3. It works on number 4. It works on number 5. It works on number 6. Oh, I got it, that's my solution."

Simulated solution time = 120 sec.

One accepted technique of comparing human and machine protocols is by Turing's Test. We converted the generation sequence in the protocols into a form in which grammar could not be a clue to the speaker's identity. A group of college students was then asked to separate the human protocols from the nonhuman. Such a procedure with 19 judges yielded a hit rate of only 52%.

While the goodness-of-fit data provided above suggest that our process model is a good predictor of human behavior, a critic might argue that we have enough free parameters to make such a fit possible even if humans employed processes that were quite different. One way to refute such an argument is to test the model's predictiveness on a new set of problems for a new set of subjects *without changing any parameters.* This is a very rigorous test of a model. In principle, to obtain a good fit our estimate of the time parameters must be good estimates of the population's time parameters.

PREDICTING BEHAVIOR WITH THE MODEL

To provide as rigorous a test as possible of the theory, we devised three new induction problems whose solutions, we knew, could be found by the program. The correct solutions to the three problems were respectively: b/a, $a^b + b$, and $a^2 - b^2$. The INDUCT-1 simulation program was tested on these problems for 10 trials per problem. The *same values of all its parameters* were used (including times) as had been used in the previous simulation runs. We then conducted a second experiment in which we tested subjects on these three new problems.

Experiment II

Subjects. The subjects were eight undergraduates from the same population as that used in Experiment I.

Procedure. This second experiment employed the same basic procedure as Experiment I with the following exceptions:

(*a*) Each of the eight new subjects was tested on three new problems in a counterbalanced design; (*b*) each of three new problems was written on a blackboard as six instances of the form $f(a,b) = c$, for example, $f(2,3) = 8$; (*c*) subjects were allowed only 15 minutes in which to solve a problem. For each problem this procedure yielded eight solution times.

Results. A comparison of the predicted

and observed solution times is shown in Figure 3. While the two curves are slightly separated, the pattern of predicted and observed times are quite similar. Considering that the second experiment was conducted by a different experimenter on different subjects with a slightly different procedure, the predictiveness of the model seems to be quite good.

The sensitivity of the model to the ordering of the data for a problem can be seen by comparing the solution times it yielded above for Problem 2 with the solution times it yielded for Problem 2 when the order of the six data instances was reversed. As shown in Table 5, the reversed order for Problem 2 placed the three instances that strongly suggest alternative hypotheses (e.g., $A + B$, $A * B + 1$) at the top. The model assumes (based on the protocols) that subjects process the data from the top down even though all the data is exposed simultaneously. Hence, the INDUCT-1 program generated solutions for the reversed order in a mean (simulated) time of 650 seconds compared with 329 seconds for the original order. But what about human subjects? Four subjects were given the problem in the reversed order and their mean time was 400 seconds compared with 291 seconds for the four who solved it in the original order.

Discussion. Our model has ignored some potential sources of information that might make even more accurate predictions possible. For example, there was evidence in the protocols that a few subjects infrequently did make use of information in the data instances to guide their search for the formula. One subject remarked that "output values are quite high with small increases in the input values; so I'll concentrate on exponentiation." In addition, the order in which subjects generated hypotheses on one problem was influenced by what they had found useful on the previous problem. For example, in Experiment II the subjects who solved Problem 2 before Problem 3 were 360 seconds faster on Problem 3, while those who solved Problem 3 first were 214 seconds faster on Problem 2. These differences are quite probably due to the fact that the correct functions for both problems involve exponentiation and addition or subtraction, that is, $A^B + B$ for Problem 2 and $A^2 - B^2$ for Problem 3.

Expanding the model to incorporate these problem-dependent features could improve its predictiveness. Yet the model's accuracy without them indicates the extent to which the generate-and-test process used to induce functions is controlled by problem-independent heuristics.

TABLE 5

COMPARISON OF OBSERVED AND PREDICITED SOLUTION TIMES FOR PROBLEM 2 WITH TWO DIFFERENT ORDERINGS OF ITS INSTANCES

	Normal ordering:	Inverted ordering:
Mean time in seconds	$f(3, 3) = 30$ $f(2, 4) = 20$ $f(4, 2) = 18$ $f(1, 1) = 2$ $f(1, 0) = 1$ $f(1, 2) = 3$	$f(1, 1) = 2$ $f(1, 0) = 1$ $f(1, 2) = 3$ $f(3, 3) = 30$ $f(2, 4) = 20$ $f(4, 2) = 18$
Predicted $N = 10$	328.6	650.0
Observed $N = 4$	291.0	400.0

Note. All instances were revealed to the subject simultaneously.

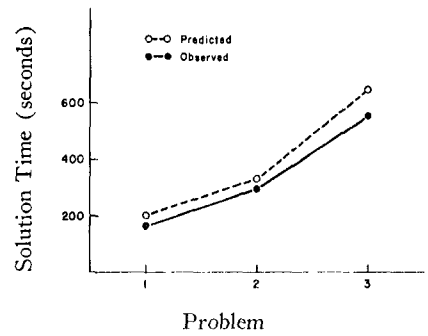

FIG. 3. A comparison of the predicted and observed solution times for the three problems used in Experiment II. (The values used for the parameters of the model were the values determined in the first simulation.)

IMPLICATIONS OF THE MODEL

The major implication of this study is that humans induce simple mathematical

functions by a process that is neither mysterious nor complicated. They generate hypotheses from predetermined pools without replacement. The pools are ordered in accord with a few heuristics independently of the data. Among the most important heuristics is that hypotheses with i operators are generated before those with more than i operators. Once generated, an hypothesis is tested until it is negated. However, negated hypotheses that have been supported by some data may be periodically retested.

While a few readers may have difficulty accepting the central thesis of this model—that induction, the major process in scientific inference, is primarily a generate-and-test process—the conclusion is consistent with most of the other research on induction in humans. One should not be surprised by this assertion unless one believes that men are above the laws of logic that govern every automaton's behavior. Nor should one conclude from this finding that men are blind and unintelligent induction machines. Intelligence in induction can only mean that the problem solver heuristically directs his generate-and-test process to find a solution rapidly.

While few data-dependent heuristics were employed on the problems studied, good data-independent heuristics were used extensively. What evidence is there that more complex heuristics would have yielded quicker solutions to these problems? One cannot conclude that data-dependent heuristics would not be used on more difficult problems because they were not used on the studied problems. On other tree-searching problems, for example, games, the findings have been that humans only employ complex heuristics when the tree becomes too massive for simpler techniques. When that point is reached, however, humans wield quite sophisticated heuristic generate-and-test techniques as has been amply demonstrated in studies of chess and other games (Newell & Simon, 1972).

With the appropriate heuristic techniques, a problem solver need not enumerate all or even a major portion of the tree or hypotheses before finding the solution. To one familiar with the recent progress in artificial intelligence, it seems quite reasonable that the major inductive feats of man could be accomplished by an automaton using a heuristically directed generate-and-test process.

Perhaps a more important criticism of man's inductive ability raised by this model is its assertion that he holds on to a theory disconfirmed by negative evidence if it has been confirmed by positive evidence. This tendency has been observed before (Wason, 1968) and appears to pervade even the most sophisticated types of inductive reasoning. As Kuhn (1970) has pointed out, the history of science is rife with cases where anomalies have not led to the rejection of a theory. Old theories are usually rejected only when a new theory is devised.

Why should man possess such inefficient persistence? Part of the reason may lie in man's recognition that he is not an errorless processor and should not trust a single disconfirming computation. The role of noise in the world may add to man's skepticism. Scientists certainly are aware that any single test can be erroneous. One might also argue that it is better to maintain a theory that brings some order to the data than to have no theory at all. Yet anyone who studies the protocols showing how subjects time and again fixate on incorrect functions that fit part of the data must question the adequacy of the above explanations.

Without rejecting any of the above theories we would like to suggest that a subject's persistence in maintaining a false hypothesis may also be due to his misperception of the probability that a false hypothesis could be confirmed. In particular, we propose that the a priori probability of a confirmation of a false hypothesis is perceived by most subjects to be much lower than it is. As a result, the decreases in subjective probability (of a hypothesis being correct) caused by a disconfirmation are erroneously offset by the increases that prior confirmations produced. For example, assume that the a priori probability that some hypothesis is correct is perceived to be .50, and the a priori probability of a confirmation of a correct hypothesis is presumed to be .90; then, if the a priori probability of a confirmation of an

incorrect hypothesis were viewed as only .10, the computed probability that some hypothesis is correct following two confirmations and one disconfirmation would be (according to Bayes' law) .90. On the other hand, if the a priori probability of a confirmation of an incorrect hypothesis were perceived (more realistically) to be .50, the computed probability of correctness after two confirmations and one disconfirmation would be only 39. Thus, too low an a priori estimate of the likelihood that a false hypothesis could be confirmed can lead a subject to believe that a negated hypothesis has a higher probability of being correct than one that has not been tried.

CONCLUSIONS

We have presented a detailed process model of how humans find mathematical functions that explain observed data. It was suggested that this model could serve as a formalization of the inductive process used in learning strategies for solving problems.

In accord with previous research on the attainment of Boolean concepts, it was found that mathematical functions are induced primarily through a process of generating hypotheses and testing them on the observed data until they are negated. The generation order is mostly independent of the characteristics of the problem's data, but the order can be specified closely with a few heuristics, for example, hypotheses with i operators are generated before those with $i + 1$ operators.

Subjects test each hypothesis on the data in a relatively fixed sequence beginning with the instance that negated the previous hypothesis or the initial instance in the sequence. A false hypothesis which matches several of the instances is often repeatedly considered even though it has been negated. The INDUCT-1 computer simulation model, incorporating these processes, accurately predicted solution times for subjects on both the set of problems on which it had been designed and a new set of problems *without any changes in its parameters.* In addition, the program's solution protocols were indistinguishable from human protocols.

REFERENCES

BONGARD, M. *Pattern recognition.* New York: Spartan, 1970.

BOWER, G., & TRABASSO, T. Concept identification. In R. C. Atkinson (Ed.), *Studies in mathematical psychology.* Stanford: Stanford University Press, 1964.

BUCHANAN, B. G., SUTHERLAND, G. L., & FEIGENBAUM, E. A. Heuristic DENDRAL: A program for generating explanatory hypothesis in organic chemistry. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence 4.* Edinburgh: Edinburgh University Press, 1969.

DUNCAN, C. P. Induction of a principle. In C. P. Duncan (Ed.), *Thinking: Current experimental studies.* New York: Lippincott, 1967.

FELDMAN, J. Some decidability results on grammatical inference and complexity. *Information and control,* 1972, **20**, 244–262.

FELDMAN, J., TONGE, F. M., JR., & KANTER, H. Empirical explorations of a hypothesis-testing model of binary choice behavior. In A. C. Hoggatt & F. E. Balderston (Eds.), *Symposium on Simulation Models.* Cincinnati, Ohio: South-Western, 1963.

GREGG, L. W. Internal representation of sequential concepts. In B. Kleinmuntz (Ed.), *Concepts and the structure of memory.* New York: Wiley, 1967.

GREGG, L. W., & SIMON, H. A. Process models and stochastic theories of simple concept formation. *Journal of Mathematical Psychology,* 1967, **4**, 246–276.

HUNT, E. B., MARIN, J., & STONE, P. J. *Experiments in induction.* New York: Academic Press, 1966.

KUHN, T. S. *The structure of scientific revolution.* Chicago: University of Chicago Press, 1970.

LEEUWENBERG, E. L. L. Quantitative specification of information in sequential patterns. *Psychological Review,* 1969, **76**, 216–220.

MINSKY, M., & PAPERT, S. *Perceptrons.* Cambridge: M.I.T. Press, 1969.

NEWELL, A., & SIMON, H. A. GPS, A program that simulates human thought. In H. Billings (Ed.), *Lernende Automaten.* Muchen: Oldenborg, 1961.

NEWELL, A., & SIMON, H. A. *Human problem solving.* Englewood Cliffs: Prentice Hall, 1972.

NILSSON, N. J. *Learning machines.* New York: McGraw-Hill, 1965.

PIVAR, M., & FINKELSTEIN, M. Automation, using LISP, of inductive inference on sequences. In E. C. Berkeley & D. G. Bobrow (Eds.), *The programming language LISP.* Cambridge, Mass.: Information International, Inc., 1964.

PLOTKIN, G. D. A further note on inductive generalization. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence 6.* New York: American Elseiver, 1971.

RESTLE, F. A. The selection of strategies in cue

learning. *Psychological Review,* 1962, **69**, 320–342.

RESTLE, F. Theory of serial pattern learning: Structural trees. *Psychological Review,* 1970, **77**, 481–495.

SIMON, H. A. Complexity and the representation of patterned sequences of symbols. *Psychological Review,* 1972, **79**, 369–382.

SIMON, H. A., & KOTOVSKY, K. Human acquisition of concepts for sequential patterns. *Psychological Review,* 1963, **70**, 534–546.

SIMON, H. A., & SUMNER, R. K. Pattern in music. In B. Kleinmuntz (Ed.), *Formal representation of human judgment.* New York: Wiley, 1968.

SMEDSLUND, J. The concept of correlation in adults. In C. P. Duncan (Ed.), *Thinking: Current experimental studies.* New York: Lippincott, 1967.

WASON, P. C. "On the failure to eliminate hypotheses . . ."—A second look. In P. C. Wason & P. Johnson-Laird (Eds.), *Thinking and reasoning.* Baltimore: Penguin, 1968.

WILLIAMS, D. S. Computer program organization induced from problem examples. In H. A. Simon & L. Siklossy (Eds.), *Representation and meaning.* Englewood Cliffs, N. J.: Prentice-Hall, 1972.

(Received July 24, 1972)

Erratum

On the front cover of the January 1973 issue (Vol. 80, No. 1) there was a typographical error in the title of the Theoretical Note by Peter H. Schönemann, Thomas Cafferty, and James Rotton. The correct title is "A Note on Additive Functional Measurement." Sorry!