PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES

Deductive Learning [and Discussion]

L. G. Valiant and J. C. Shepherdson

Phil. Trans. R. Soc. Lond. A 1984 **312**, 441-446
doi: 10.1098/rsta.1984.0069

Phil. Trans. R. Soc. Lond. A **312**, 441–446 (1984) [441]
Printed in Great Britain

Deductive learning

By L. G. Valiant
Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138, *U.S.A.*

A non-technical discussion of a new approach to the problem of concept learning in the context of artificial devices is given. Learning is viewed as a process of acquiring a program for recognizing a concept from an environment that does not reveal an explicit description of the program but only suggests it by such means as identifying positive examples of it. The proposed model makes possible a study of learning that reconciles three requirements: the classes of concepts that can be learnt are relevant for general purpose knowledge; they can be characterized; the process of learning them is computationally feasible.

1. Introduction

We consider the following computational model of knowledge. A robot receives information from the outside world via a set of primitive Boolean variables $x_1, \ldots, x_n$. Its task is to recognize whether various predicates or *concepts* are exemplifed in the information presented. The knowledge base of the robot is a set of programs $f_1, \ldots, f_m$ that compute Boolean functions $F_1 \ldots, F_m$ respectively, one corresponding to each concept. These programs may use each other as subroutines. They can take as input parameters both the primitive variables and the output of other programs.

The question we ask is: how can the robot acquire a program for a further function F_{m+1} if no source for providing an explicit listing of that program is available? In this paper we describe such a process of *program acquisition without being programmed* as *learning*. This is a central issue in artificial intelligence. A primary goal there is to provide machines with human-like skills for which satisfactory algorithms are often unknown. Even when such algorithms are known the problem of adding one to a knowledge base that is already very complex and difficult to understand may present difficulties.

Our purpose here is to formulate and discuss a rigorous approach to understanding such learning processes. A first aim would be to delimit the class of functions that are learnable from those that are not. In this model the impediment to learnability is computational complexity. If members of a class of programs can be acquired by learning only in exponentially many steps then this class will not be learnable in practice. On the positive side we expect that the model will provide particular proposals for realistic learning systems. A more complete treatment of our approach appears elsewhere (Valiant 1984).

Learning is often associated with the notion of induction. This association emphasizes the fact that if we are deriving a general principle from a limited number of examples, say, then there is, inevitably, some element of inspiration or guesswork involved, since the information available is simply insufficient for a deduction to be made with certainty.

In our title we use the word deductive to emphasize the contrary point. Human learning often shows remarkable properties of convergence. In large populations there is a high degree

of agreement on the meaning of thousands of words as they relate to everyday situations. This suggests that highly reliable program acquisition may be feasible even in the absence of explicit programming. Hence it is reasonable to insist that our study of learning be disciplined by insistence on fast convergence.

One motivation of this study is encapsulated in the informal notion of an 'optimal learning situation'. Here a robot eager to learn, already possessing much knowledge as well as the best general purpose learning strategies, wishes to learn its next new concept from a teacher that understands the concept well and is willing to go to any lengths to impart to the robot a recognition algorithm for the concept, short of providing an explicit description of such a program.

To give a precise meaning to the above we have to define the *learning protocol*, the manner of interaction that is allowed between the learner and teacher. This will typically include the teacher making positive or negative identifications of the concept for various inputs, and, possibly, the teacher answering questions posed by the learner. We call this study that of *learnability* because its aims include that of understanding the maximal limits of the learning phenomenon that are feasible with reasonable learning protocols.

To represent knowledge and programs we shall use propositional calculus expressions. On the one hand they are so simple and central in knowledge representation that it is difficult to imagine how the learning issue can be understood without considering them. As we shall see, however, the task of learning even in this limited context poses serious problems. It would appear to be over ambitious to attempt much more before the questions raised here are better understood.

The study of learning that we suggest is, in the first instance, a theoretical one in the spirit of, say, computability theory. Whether it will ever be efficacious to make artificial systems with general purpose learning capabilities depends on whether efficient strategies exist and can be discovered for classes of programs that are substantial enough. It may turn out that even if very good strategies exist, they only start being usable when a very large knowledge base has been built. In that case the validation of such strategies for practical purposes would involve major challenges for the experimenter.

Extensive bibliographies on previous work on inductive inference and machine learning are given by Angluin & Smith (1982) and Michalski *et al.* (1983).

2. A problem of formulation

We consider propositional expressions in a set of propositional variables $p_1, \ldots, p_t$. An extremely simple kind of expression is a product or *monomial*. For example, $f = p_2 \bar{p}_4 p_5$ is a monomial. It takes the value 'true' or '1' if p_2 and p_5 are both true and p_4 false, whatever values the other variables take. The expression f is 'false' if p_2 is false or if p_5 is false or if p_4 is true or if any combination of these conditions holds.

It is realistic to consider that t is very large. A system may have large numbers of primitive inputs and of functions that are programmed or previously learnt. Each single function may be dependent on only a small fraction of the t variables.

A *positive example* of a function or expression will be a vector of truth assignments to the variables that makes the function or expression true. Thus $(p_1 = 1, p_2 = 1, p_4 = 0, p_5 = 1)$ will be a positive example of the particular expression f but $(p_2 = 0, p_4 = 0, p_5 = 1, p_6 = 1)$ and

($p_2 = 1$, $p_4 = 0$) will not be. The omission of a variable p_i, or equivalently setting $p_i = *$, denotes that the value of p_i is *undetermined*. It is important to allow for undetermined variables because in realistic situations confirmations of a concept may be obtained from the values of a small set of the variables. The values of some of the remaining variables may not be obtainable. For example, in recognizing an 'elephant' the colour, size, location, etc., may all be relevant but it may be possible to make a positive identification even when the values of some of the variables are unknown.

Suppose that a robot is presented with a large number N of vectors that are identified as positive examples of f. The task of the robot is to deduce reliably an expression that equals or in some sense approximates f. The question of defining a plausible learning mechanism even for such apparently trivial expressions as monomials is problematic and was previously unsolved. The difficulty lies in giving a definition of learning that is both realistic and computationally feasible.

A first attempt at a definition may be to insist that f be deducible from any set of N distinct positive examples. This, however, is unrealistically optimistic and may require N to be exponentially large in terms of t. For example, suppose that $f = p_1$ say and $N = 2^{\frac{1}{2}t}$; then an unfortunate choice of examples would assign $p_1 = p_2 = \ldots = p_{\frac{1}{2}t} = 1$ and would vary over all combinations of assignments to the remaining variables. These examples clearly reveal little about the nature of f.

A second attempt may be to define some natural probabilistic distribution for the relative probability of occurrence of the various vectors. For example, we could say that all the 2^{t-1} vectors consistent with $f = p_1$ have the same probability of occurring. This model would make unfortunate choices of examples unlikely and would therefore solve the problem technically. Unfortunately in the real world this assumption will nearly always be totally false. The actual distributions for the various concepts of interest may bear no relation to each other and will not be known *a priori* in general.

3. A probabilistic model

The solution proposed to the above predicament is a very simple one: assume that vectors of variable values do occur in nature with some fixed probabilistic distribution. Allow this distribution, however, to be completely *arbitrary* and unknown. Furthermore require of the learner only the ability to deduce an expression that approximates f in that if a vector is drawn randomly from the distribution then the deduced expression may disagree with f on this vector, but the probability of this happening is small. Thus, pursuing the example $f = p_1$ of the previous section, suppose that $p_2 = 1$ in 99% of positive examples of f. Then a learner who has not seen a positive example with $p_2 = 0$ or p_2 undetermined, may deduce that $f = p_1 p_2$. Our definitions tolerate this inaccuracy because the learner's program $p_1 p_2$ will in that case indeed be equivalent to p_1 for 99% of the time in real-world examples of the concept.

The deduction algorithm for the expression we assume above is a simple one: let the deduced expression be the product of all p_i or $\bar{p}_i$ that are determined to be true in *all* the positive examples seen by the learner.

Now without knowing anything about the distribution the following is provable. For any number $h > 1$, if $2h(t + \log_2 h)$ positive examples are drawn from the distribution, then with probability at least $1 - h^{-1}$ the deduced expression g will have the property: (i) for any vector

v for which g is true f is also true; (ii) if a vector v for which f is true is drawn at random with probability determined by the distribution of such positive examples of f, then v will make g true with probability at least $1-h^{-1}$.

The possibility of proving such results suggests the general definition of what it means for a class of expressions, or more generally programs, to be learnable. Suppose that X is a class of programs. For a typical member $f \in X$ we denote the size of f by the parameter s, and its arguments by $p_1, \ldots, p_t$. We denote an assignment of truth values to $p_1, \ldots, p_t$ by the vector $v \in \{0, 1, *\}^t$. We denote by D and $\bar{D}$, respectively, the arbitrary probability distributions over $\{v|f \text{ is true on } v\}$ and $\{v|f \text{ is not true on } v\}$. Thus D(v) denotes the relative probability of occurrence of v among vectors that make f true. $\bar{D}(v)$ is the corresponding distribution for the complement set. Note that if f is not true on input v it may be either false or undefined.

The class X is *learnable* with respect to a learning protocol L if there is a deduction procedure using L such that

(i) the procedure runs in time polynomial in s, t and in an adjustable parameter h, and

(ii) for all $f \in X$ and all D, $\bar{D}$ the algorithm will deduce, with probability at least $1-h^{-1}$, a program $g \in X$ having the properties:

(*a*) $\Sigma\{D(v)|f \text{ is true but g is not true on } v\} \leqslant h^{-1}$;

(*b*) $\Sigma\{\bar{D}(v)|f \text{ is not true but g is true on } v\} \leqslant h^{-1}$.

In the example cited earlier the learning protocol consists of a supply of positive examples. The learner is allowed to call a procedure EXAMPLES, which always returns the value of a vector v such that f is true on v. The probability that a particular v is provided is exactly D(v).

The definition given of learnability allows for *two-sided error*. In the example we gave of learning single monomials, it is clear that if $f(v) \neq 1$ it is impossible that $g(v) = 1$. In this case, and in all the other cases so far considered (Valiant 1984), learning can be achieved with only *one-sided error*. Learnability with one-sided error is defined in the same way as for two-sided error, except that condition (ii) (*b*) is replaced by 'for all v if g is true on v then so is f'. This is an important advantage because it allowed us to omit mentioning $\bar{D}$. While it may be reasonable to discuss the distribution of the attributes of elephants, we may prefer not discussing the distribution of the attributes of non-elephants.

The most interesting class of propositional expressions that we can show to be learnable (with one-sided error) is the class of k-CNF *expressions* with at most k literals (negated or unnegated variables) in each clause. For example $(p_1+\bar{p}_3)(p_2+p_3)(p_2+\bar{p}_3)$ is a 2-CNF expression. 3-CNF expressions are already complex in the sense that it is NP-hard to determine whether such an expression has the value zero for all vectors v. Nevertheless the following theorem can be proved.

THEOREM 1. *For every positive integer* k *the class of* k-CNF *expressions is learnable with respect to the positive examples learning protocol.*

The expressions that appear to be easiest to understand for humans are disjunctive normal form (or sum of product) expressions, such as $p_1 p_3 p_4 + p_2 p_4$. An expression is *monotone* if no variable occurs negated. A very immediate question that is currently unresolved is whether monotone DNF expressions are learnable with respect to the positive examples protocol.

4. Oracles

The idea of learning from positive or negative examples is appealing because the teacher has only the minimal role of making the identifications. Unfortunately the classes of expressions that we can show to be learnable from examples alone are very limited. It is therefore necessary to ask how we can formulate more active forms of teaching in the context of an optimal learning situation. The solution we propose is that of introducing various *oracles*. Each oracle can answer questions of a specified nature about the concept to be learnt. The learning protocol allows the learner to pose questions and receive answers to them.

A simplest oracle is one of *necessity*. Given a vector of truth assignments to a subset of the variables it will determine whether the vector is a positive example or not. Note that if a vector is a positive example then so will also any vector obtained from it by making some undetermined variables determined. The power of such an oracle is illustrated by the fact that if the learner has to decide whether the correct expression sought is p_1 or $p_1 p_2$ it can simply input '$p_1 = 1$' to the oracle and will get a positive answer if the correct formula is p_1, and a negative one if it is $p_1 p_2$. The following theorem can be proved.

Theorem 2. *The class of monotone disjunctive normal form expressions is learnable with respect to the protocol that allows positive examples and the necessity oracle.*

The size measure in theorem 2 is the number of symbols in the expression. When negations are allowed, if the size measure is redefined appropriately a similar result can be proved. An interesting open question is whether monotone CNF expressions with no bound on clause size are learnable via positive examples and the necessity oracle.

One can define a variety of other oracles for kinds of questions that a person understanding a concept may be expected to be able to answer. For example a *possibility* oracle would tell given a vector whether it is possible to make the expression true by giving the undetermined variables some suitable values. Thus the assertion 'large = 1' is a vector for which it is possible but not necessary that the described thing is an elephant. An oracle of *relevant possibility* would tell, given a vector, whether by making determined some undetermined variables, one can obtain a vector that is (necessarily) a positive example, but such that making any determined variable undetermined in it would not give a (necessarily) positive example. Thus 'large = 1' is relevantly possible if in *some* conjunctive criterion for elephants the question of largeness cannot be dropped.

One can say informally that an oracle is reasonable if it gives insight into the learning process. Oracles that give explicit answers about the syntactic description of the program to be learnt are not reasonable because they are thinly disguised programming languages.

5. Limits of learnability

It is conceivable that all programs, when described as Boolean circuits, say, are learnable. There is, however, substantial circumstantial evidence from cryptography that this is not so and that the class of learnable programs is very restricted. One task of cryptography is to find encoding schemes such that an enemy that has access to even a large sample of previous messages and their encodings is unable to replicate the encoding algorithm. Our notion of learnability, at least when restricted to protocols using examples alone, requires the converse property. We

PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES

require that from the input–output behaviour of any member of a class of programs the particular program be easily replicated. Furthermore in encoding schemes no member of the class should be deducible, while in a learnable class we expect them all to be deducible. Hence the widely conjectured existence of encoding schemes that are computationally simple suggests that learnable classes of programs may be extremely limited.

In the light of the above observation the question arises as to how the acquisition of complex programs or skills should be viewed. The view we propose is that while simple programs can be learnt, in our sense, in anything more complicated a programming element becomes essential. It is conceivable, for example, that given the state of knowledge and learning strategies of a particular robot the concept of 'elephant' is not learnable for it. It may be, however, that the notion of 'trunk' is learnable and that, once that concept is learnt, 'elephant' becomes learnable also. Such intervention by the teacher in identifying and sequencing intermediate concepts we classify as programming.

This research was supported in part by National Science Foundation Grant MCS-83-02385.

References

Angluin, D. & Smith, C. H. 1982 A survey of inductive inference: theory and methods. *Yale University, Computer Science Department Tech. Rep.* no. 250.

Michalski, R. S., Carbonell, J. G. & Mitchell, T. M. 1983 *Machine learning: an artificial intelligence approach.* Palo Alto, California: Tioga.

Valiant, L. G. 1984 A theory of the learnable. In *Proc. 16th Ass. Comput. Mach. Symp. on Theory of Computing*, pp. 436–445. New York: Association for Computing Machinery.

Discussion

J. C. Shepherdson (*School of Mathematics, University of Bristol, U.K.*). Professor Valiant emphasized that he was talking about *deductive inference*. Do his results apply also to other kinds of inference, for example to machines that make random guesses on the way to fixing on a program? Could he argue that they do, because a random element could always be replaced by a pseudo-random element that could be computed in polynomial time?

L. G. Valiant. The word *deductive* was intended to emphasize that the formulation constrains the learning process to converge quickly and in a demanding manner. The formulation contains a probabilistic element arising from the probabilistic distribution on the example space and, as it happens, already allows for the possibility that additionally there is randomizaton in the learning procedure itself. Although I have not found an application for this idea it is certainly worth pursuing. Whether true randomness can be replaced by pseudo-randomness in the context of polynomial time algorithms is still an open problem.

PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES