Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Algorithmic identification of probabilities is hard

Laurent Bienvenu [a,*], Santiago Figueira [b], Benoit Monin [c], Alexander Shen [a]

[a] LIRMM, CNRS & Université de Montpellier, 161 rue Ada, 34095 Montpellier Cedex 5, France
[b] Universidad de Buenos Aires and CONICET, Pabellón I – Ciudad Universitaria, Buenos Aires, Argentina
[c] LACL, Université Paris 12, 61 avenue du Général de Gaulle, 94010 Créteil Cedex, France

ARTICLE INFO

Article history:
Received 12 April 2017
Received in revised form 28 December 2017
Accepted 10 January 2018
Available online xxxx

Keywords:
Algorithmic learning theory
Algorithmic randomness

ABSTRACT

Reading more and more bits from an infinite binary sequence that is random for a Bernoulli measure with parameter p, we can get better and better approximations of p using the strong law of large numbers. In this paper, we study a similar situation from the viewpoint of inductive inference. Assume that p is a computable real, and we have to eventually guess the program that computes p. We show that this cannot be done computably, and extend this result to more general computable distributions. We also provide a weak positive result showing that looking at a sequence X generated according to some computable probability measure, we can guess a sequence of algorithms that, starting from some point, compute a measure that makes X Martin-Löf random.

1. Introduction

1.1. Inductive inference

The study of learnability of computable sequences is concerned with the following problem. Suppose we have a black box that generates some infinite computable sequence of bits $X = X(0)X(1)X(2), \ldots$ We do not know the program running in the box, and want to guess it by looking at finite prefixes

$$X\restriction n = X(0)\ldots X(n-1)$$

for increasing values of n. There could be different programs that produce the same sequence, and it is enough to guess one of them (since there is no way to distinguish between them by just looking at the output bits). The more bits we see, the more information we have about the sequence. For example, it is hard to say something about a sequence seeing only that its first bit is a 1, but looking at the prefix

11001001000011111101101010100 0

one may observe that this is a prefix of the binary expansion of π, and guess that the machine inside the box does exactly that (though the machine may as well produce the binary expansion of, say, 47627751/15160384).

* Corresponding author.
E-mail address: laurent.bienvenu@computability.fr (L. Bienvenu).

https://doi.org/10.1016/j.jcss.2018.01.002
0022-0000/

Please cite this article in press as: L. Bienvenu et al., Algorithmic identification of probabilities is hard, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.01.002

The hope is that, as we gain access to more and more bits, we will *eventually* figure out how the sequence X is generated. More precisely, we hope to have a total computable function $\mathfrak{A}$ from strings to integers such that for every computable X, the sequence

$$\mathfrak{A}(X\upharpoonright 1), \mathfrak{A}(X\upharpoonright 2), \mathfrak{A}(X\upharpoonright 3), \dots$$

converges to a program (= index of a computable function) that computes X. This is referred to as *identification in the limit*, and can be understood in (at least) two ways. Indeed, assuming that we have a fixed effective enumeration $(\varphi_e)_{e\in\mathbb{N}}$ of partial computable functions from $\mathbb{N}$ to $\{0,1\}$, we can define two kinds of success for an algorithm $\mathfrak{A}$ on a computable sequence X:

- Strong success: the sequence $e_n = \mathfrak{A}(X\upharpoonright n)$ converges to a single value e such that $\varphi_e = X$ (i.e., $\varphi_e(k) = X(k)$ for all k).
- Weak success: the sequence $e_n = \mathfrak{A}(X\upharpoonright n)$ does not necessarily converge, but $\varphi_{e_n} = X$ for all sufficiently large n.

Here we assume that $\mathfrak{A}(X\upharpoonright n)$ is defined for all n or at least for all sufficiently large n.

The strong type of success is often referred to as *explanatory* (EX), see, e.g., Definition VII.5.25 in [7, p. 116]. The second type is referred (see Definition VII.5.44, p. 131 in the same book) as *behaviorally correct* (BC). Note that it is obvious from the definition that strong success implies weak success.

It would be nice to have an algorithm that succeeds on all computable sequences. However, it is impossible even for weak success: for every (total) algorithm $\mathfrak{A}$, there is a computable X such that $\mathfrak{A}$ does not weakly succeed on X. The main obstacle is that certain machines are not total (produce only finitely many bits), and distinguishing total machines from non-total ones cannot be done computably.

However, some *classes* of computable sequences can be learned, i.e., there exists a total algorithm that succeeds on all elements of the class. Consider for example the class of primitive recursive functions. This class can be effectively enumerated, i.e., there is a total computable function f such that $(\varphi_{f(e)})_{e\in\mathbb{N}}$ is exactly the family of primitive recursive functions. Now consider the algorithm $\mathfrak{A}$ such that $\mathfrak{A}(\sigma)$ returns the smallest e such that $\varphi_{f(e)}(i) = \sigma(i)$ for all $i < |\sigma|$ (such an e always exists, since every string is a prefix of a primitive recursive sequence). It is easy to see that if X is primitive recursive, $\mathfrak{A}$ succeeds on X, even in the strong sense (EX).

The theory of learnability of computable sequences (or functions) is precisely about determining which classes of functions can be learned. This depends on the learning model, the type of success, of which there are many variants. We refer to the survey by Zeugman and Zilles [11] and to [7, Chapter VII] for a panorama of the field.

1.2. Learning measures

Recently, Vitányi and Chater [9] proposed to study a related problem. Suppose that instead of a sequence that has been produced by a deterministic machine, we are given a sequence that has been generated by a randomized algorithmic process, i.e., by a Turing machine that has access to a fair coin and produces some output sequence on the one-directional write-only output tape. The output sequence is therefore a random variable defined on the probabilistic space of fair coin tossings. We assume that this machine is *almost total*.[1] This means that the generated sequence is infinite with probability 1.

Looking at the prefix of the sequence, we would like to guess which machine is producing it. For example, for the sequence

$$0001111111100001100000000001111111111111$$

we may guess that it has been generated via the following process: start with 0 and then choose each output bit to be equal to the previous one with probability, say, $4/5$ (so the change happens with probability $1/5$), making all the choices independently.[2]

So what should count as a good guess for some observed sequence? Again, there is no hope to distinguish between two processes that have the same output distribution. So our goal should be to reconstruct the output distribution and not the specific machine.

But even this is too much to ask for. Assume that we have agreed that some machine M with output distribution μ is a plausible explanation for some sequence X. Consider another machine M' that starts by tossing a coin and then (depending on the outcome) either generates an infinite sequence of zeros or simulates M. If X is a plausible output of M, then X is also a plausible output of M', because it may happen (with probability $1/2$) that M' simulates M.

A reasonable formalization of a 'good guess' is provided by the theory of algorithmic randomness. As Chater and Vitányi recall, there is a widely accepted formalization of "plausible outputs" for an almost total probabilistic machine with output distribution μ: the notion of Martin-Löf random sequences with respect to μ. These are the sequences that pass all effective

[1] This requirement may look unnecessary. Still the notion of algorithmic randomness needed for our formalization is well-defined only for computable measures, and machines that are not almost total may not define a computable measure.

[2] The probability $4/5$ is not a dyadic rational number, but still can be simulated by an almost total machine using a fair coin.

Please cite this article in press as: L. Bienvenu et al., Algorithmic identification of probabilities is hard, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.01.002