Theoretical Computer Science 620 (2016) 33–45

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Topological separations in inductive inference

John Case [a], Timo Kötzing [b,*]

[a] Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA
[b] Department 1: Algorithms and Complexity, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Available online 30 October 2015

Keywords:
Inductive inference
Language learning
Non-computable learning
Topological

ABSTRACT

Re learning in the limit from positive data, a major concern is which classes of languages are learnable with respect to a given learning criterion. We are particularly interested herein in the reasons for a class of languages to be *un*learnable. We consider two types of reasons. One type is called *topological* where it does not help if the learners are allowed to be *un*computable (an example of Gold's is that no class containing an infinite language and all its finite sub-languages is learnable — even by an uncomputable learner). Another reason is called *computational* (where the learners are required to be algorithmic). In particular, two learning criteria might allow for learning different classes of languages from one another — *but* with dependence on whether the unlearnability is of type topological or computational.

In this paper we formalize the idea of two learning criteria *separating topologically* in learning power. This allows us to study more closely why two learning criteria separate in learning power. For a variety of learning criteria, concerning vacillatory, monotone, (several kinds of) iterative and feedback learning, we show that certain learning criteria separate topologically, and certain others, which are known to separate, are shown *not* to separate topologically. Showing that learning criteria do not separate topologically implies that any known separation must necessarily exploit algorithmicity of the learner.

1. Introduction

The learning theory of this paper pertains to trial and error learning of (algorithmic) descriptions, i.e., grammars or programs, for formal languages L. This kind of learning is sometimes called *learning in the limit*, and herein it is such learning from positive data only re such L. The languages are taken without loss of generality to be computably enumerable sets of non-negative integers (i.e., natural numbers). As an example: a learner h (either algorithmic or not) is presented, in some order, all and only the even numbers, and, after it sees for a while only multiples of 4, it outputs some description of the set of multiples of 4. Then, when, h sees a non-multiple of 4, it outputs a description of the entire set of even numbers.

Many criteria for saying whether a learner h is *successful* on a language L have been proposed in the literature. Gold, in his seminal paper [11], gave a first, simple learning criterion, we call **TxtGEx**-*learning*,[1] where a learner is *successful* iff, on every *text* for L (a listing of all and only the elements of L), it eventually stops changing its conjectures, and its final conjecture is a correct description for L (this latter is the *explanatory* part of **TxtGEx**). Trivially, each single, describable

* Corresponding author.
E-mail addresses: case@udel.edu (J. Case), koetzing@mpi-inf.mpg.de (T. Kötzing).

[1] *Txt* stands for learning from a *text* (list) of positive examples; **G** stands for Gold, who first described this mode of learning in the limit [11]; *Ex* stands for *explanatory*.

http://dx.doi.org/10.1016/j.tcs.2015.10.036
0304-3975/

language L has a suitable constant function as an **TxtGEx**-learner (this learner constantly outputs a description for L). Thus, we are interested instead in knowing for which *classes of languages* $\mathcal{L}$ there is a *single learner* h learning *each* member of $\mathcal{L}$. A wide range of learning criteria including **TxtGEx**-learning have been investigated (see, for example, the textbook [17]).

Already Gold [11] found that certain classes of languages are not **TxtGEx**-learnable because of what was later called topological considerations,[2] e.g., when trying to **TxtGEx**-learn a class of languages containing an infinite language and all the finite subsets of it, the learner cannot distinguish between the infinite set and any of its finite subsets as, at any time, the learner has seen only finitely much positive data (and is missing information about the complement of the language); furthermore, it turns out, for this example, not even uncomputable h can learn the class. Angluin [1] described another essentially topological restriction of **TxtGEx**-learning. Intuitively, when one of these restrictions is not met, the learner just does not get enough information to be successful, regardless of its power. We collect a number of previously known topological constraints on **TxtGEx**-learning in Section 3, along with such constraints for so-called strongly monotone learning.

A lot of work re the learning theory area of the present paper centers around deciding whether one learning criterion I allows for learning classes of languages which are not learnable in another learning criterion I' (we then say that I *separates* from I'). We are interested herein in analyzing more closely the reasons for learning criteria to separate. In practice, such separations of learning criteria either involve intricate *computational* (or algorithmicity) arguments (such as program self-reference arguments or reductions to algorithmically undecidable sets) or topological arguments. We give next an example of each.

A learner is said to be *consistent* if, at any point, the language described by its conjecture at that point contains all the data known at that same point. We write consistent **TxtGEx**-learning as **TxtGConsEx** when only computable learners are considered. It is well known that **TxtGEx** separates from **TxtGConsEx** [24]. An example class that cannot be **TxtGEx**-learned consistently is the class of all non-empty languages where the least element is a coded description for the language (a numerical name of a description in some acceptable numbering of all computably enumerable sets). It is clear that the reason that this class cannot be learned consistently by a computable learner is the algorithmic undecidability of the consistency of a conjecture. And, indeed, if the learners in both criteria are not restricted to be computable, the same classes of languages are learnable.

In contrast to this, consider *iterative* learning [29,28]. At any point, an iterative learner has as its input only its just previous conjecture and the current text datum. Iterative learning proceeds by processing the text item by item and also requires the convergence to a correct conjecture (the **Ex** part), so that we call this learning criterion **TxtItEx**. It is well-known that **TxtGEx** separates from **TxtItEx**. We consider the following proof of this separation [20,22]. Let $\mathcal{L}$ be the class containing the language $\mathbb{N}^+$ (the language of all positive natural numbers) as well as every finite language containing 0. This class of languages is clearly **TxtGEx**-learnable, even by learners which map a string of inputs to a conjecture in linear time. However, this class cannot be **TxtItEx**-learned. For suppose, by way of contradiction, that some (possibly even non-computable) h would **TxtItEx**-learn this class of languages. Then, when being fed positive numbers, h will eventually output a conjecture for $\mathbb{N}^+$ and not change conjecture any more. If now, after some more positive numbers, a 0 is presented, h has "forgotten" which positive numbers were presented. A more formal proof can be found after the statement of Theorem 4.4 below. This shows how iterative learning leaves the learner at an informational disadvantage; even removing any requirement of computability for the learner cannot enable this iterative learning.

We would like to call separations of the first kind *computational*, and separations of the second kind *topological*. Note, though, that the separating class in the second/topological example can be indexed in such a way that membership in the languages in the class is uniformly decidable in *linear time*, while in the first/computational example the separating class was not a uniformly decidable class at all. Thus, we formalize our idea of topological separation versus computational separation herein as follows. We say that a learning criterion I *separates topologically* from a learning criterion I' iff there is a uniformly linear-time decidable class of languages I-learnable by a linear-time computable learner, but not I'-learnable even by non-computable learners (see Section 2 for a more formal definition). Why do we require the uniform decision procedure and the learner to be computable in linear time? There are at least two reasons. First, if a separation was witnessed only by classes which are learnable by total learners, but not computationally very simple learners, then one can hardly claim that there is no computational component to the separation; the same holds for the uniform decision procedure. Second, our separations are stronger than if we would require only uniform computability. If two learning criteria separate, but not topologically, then we say that these learning criteria *separate computationally*.

With these definitions we now have that **TxtGEx** and **TxtItEx** separate topologically, while **TxtGEx** and **TxtGConsEx** separate only computationally. However, although some uncomputable learners *can* test consistency (in general they would have to decide the halting problem) we do get that *some* learning criteria *do* separate *topologically* from their consistent variant: **TxtItEx** and **TxtItConsEx** separate topologically, as our Theorem 4.5 in Section 4 below shows.

We next summarize informally some of our other main theorems also in Section 4 below.

[2] These topological considerations arise, for example, for **TxtGEx**-learning, because learning from positive data is missing information, e.g., the negative data. They are involved in unlearnability results which hold for *all* learners h of the relevant type fitting the criterion at hand — including, in particular, all such *un*computable hs. The associated proofs of unlearnability typically feature directly or indirectly plays of a winning strategy for a Banach–Mazur game where the goal set is co-meager — as in Baire category theory [14] — and Baire category theory is part of topology. The connection to Baire category theory was first observed in [23] (see also [24]).