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.

© 2015 Elsevier B.V. All rights reserved.

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/© 2015 Elsevier B.V. All rights reserved.

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]).

For $k > 0$, **TxtGFex**$_k$-learning is just like **TxtGEx**-learning except that instead of being restricted to exactly 1 correct output conjecture in the limit, **TxtGFex**$_k$-learning allows up to k correct output conjectures in the limit. Computationally, i.e., with algorithmic learners h, from [4], these criteria form a strict learning power hierarchy with increasing k; however, surprisingly, from our Theorem 4.2 below, topological separation fails, and the hierarchy collapses when uncomputable learners are also allowed.

A *strong non-U-shaped iterative (***TxtItSNUEx***) learner* h [10] additionally satisfies: on any text T for a language L to be learned, if h on T ever outputs a correct conjecture for L, then h, forever after on T, must output this syntactically same conjecture. It is shown in [6] that **TxtItEx** separates from **TxtItSNUEx**. From Theorem 4.6 below in Section 4, this separation is even topological.

An *iterative with counter (***TxtItCtrEx***) learner* h [9] additionally has input access to the *number* of not necessarily distinct data items seen so far. In [9] it is shown that **TxtItCtrEx** separates from **TxtItEx**. Surprisingly, from Theorem 4.7 below in Section 4, this separation is actually topological.

TxtFb$_k$**Ex**-learning is just like **TxtItEx**-learning except that the learner, at any point, can, re the data presented before that point, simultaneously ask for each of up to k numbers whether it is in that prior presented data, and the learner can react to the answers. In [5] it is shown that these criteria also form a strict learning power hierarchy *computationally* with increasing k; however, surprisingly, from our Theorem 4.9 below, the hierarchy *also holds topologically*. But, from our Theorem 4.10 below, the hierarchy *collapses topologically* when the potential separation witnesses $\mathcal{L}$ are restricted to contain *no* finite languages.[3]

We believe that our work in this paper gives structural insight into learning criteria and their differences. Furthermore, topological separations embody a certain economy when showing learning criteria to separate: corollaries to each topological separation are separations with respect to learner-restricted criteria (such as partial, total or linear time computable learners) and with respect to different levels of language complexities (such as arbitrary, uniformly decidable or uniformly decidable in linear time language classes). Finally, this work also shows how a study of uncomputable learners can help understand learning with restricted computational power.

This paper is an extended version of [7].

2. Mathematical preliminaries

Unintroduced complexity theoretic notation follows [26]. Other unintroduced notation follows [27].

$\mathbb{N}$ denotes the set of natural numbers, $\{0, 1, 2, \ldots\}$. We let $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The symbols $\subseteq$, $\subset$, $\supseteq$, $\supset$ respectively denote the subset, proper subset, superset and proper superset relation between sets. $\emptyset$ denotes both the empty set and the empty sequence. $\mathfrak{R}$ denotes the set of all total functions $\mathbb{N} \to \mathbb{N}$; **LinF** is the set of all linear-time computable such functions.

With dom and range we denote, respectively, domain and range of a given function. We sometimes denote a partial function f of $n > 0$ arguments $x_1, \ldots, x_n$ in lambda notation (as in Lisp) as $\lambda x_1, \ldots, x_n . f(x_1, \ldots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x . c$ is the constantly c function of one argument.

We let $\langle \cdot, \cdot \rangle$ be a linear time computable, linear time invertible, pairing function [26] (a pairing function is a 1–1 and onto mapping $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$). Whenever we consider tuples of natural numbers as input to a function, it is understood that the general coding function $\langle \cdot, \cdot \rangle$ is used to code the tuples into a single natural number. We similarly fix a coding for finite sets and sequences, so that we can use those as input as well.

If a function f is not defined for some argument x, then we denote this fact by $f(x)\uparrow$, and we say that f on x *diverges*; the opposite is denoted by $f(x)\downarrow$, and we say that f on x *converges*. If f on x converges to p, then we denote this fact by $f(x)\downarrow = p$.

The special symbol ? is used as a possible hypothesis (meaning "no change of hypothesis"). We write $f \to p$ to denote that $f : \mathbb{N} \to \mathbb{N} \cup \{?\}$ *converges to* p, i.e., $\exists x_0 : f(x_0) = p \wedge \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$.[4] $\mathcal{P}$ and $\mathcal{R}$ denote, respectively, the set of all partial computable and the set of all computable functions (mapping $\mathbb{N} \to \mathbb{N}$).

We let φ be any fixed acceptable programming system for $\mathcal{P}$ (an acceptable programming system could, for example, be based on a natural programming language such as C or Java, or on Turing machines). Further, we let φ_p denote the partial computable function computed by the φ-program with code number p. A set $L \subseteq \mathbb{N}$ is *computably enumerable (*ce*)* iff it is the domain of a computable function. Let $\mathcal{E}$ denote the set of all ce sets. We let W be the mapping such that $\forall e : W(e) = \mathrm{dom}(\varphi_e)$. For each e, we write W_e instead of $W(e)$. W is, then, a mapping from $\mathbb{N}$ *onto* $\mathcal{E}$. We say that e is an index, or program, (in W) for W_e.

The symbol # is pronounced *pause* and is used to symbolize "no new input data" in a text. For each (possibly infinite) sequence q with its range contained in $\mathbb{N} \cup \{\#\}$, let $\mathrm{content}(q) = (\mathrm{range}(q) \setminus \{\#\})$. For any function f and all i, we use $f[i]$ to denote the sequence $f(0), \ldots, f(i-1)$ (the empty sequence if $i = 0$ and undefined, if one of these values is undefined). We denote concatenation of sequences by juxtaposition.

[3] This is interesting since most formal linguists model natural languages as infinite sets.

[4] f on x converges should not be confused with f converges *to*.

2.1. Learning criteria

In this section we formally introduce our setting of learning in the limit and associated learning criteria. We follow [18] in its "building-blocks" approach for defining learning criteria.

A *learner* is a partial function from $\mathbb{N}$ to $\mathbb{N} \cup \{?\}$ (note that, for this paper, we do not always require computability of learners). A *language* is a $\subseteq$ set $L \subseteq \mathbb{N}$. Any total function $T : \mathbb{N} \to \mathbb{N} \cup \{\#\}$ is called a *text*. For any given language L, a *text for* L is a text T such that $\text{content}(T) = L$. This kind of text is what learners usually get as information. With $\mathbf{Txt}(L)$ we denote the set of all texts for L.

A *sequence generating operator* is an operator β taking as arguments a function h (the learner) and a text T and that outputs a function p. We call p the *learning sequence* of h given T. Intuitively, β defines how a learner can interact with a given text to produce a sequence of conjectures.

We define the sequence generating operators **G** and **It** (corresponding to the learning criteria discussed in the introduction) as follows. For all learners h, texts T and all i,

$$\mathbf{G}(h, T)(i) = h(T[i]);$$

$$\mathbf{It}(h, T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0;^{5} \\ h(\mathbf{It}(h, T)(i-1), T(i-1)), & \text{otherwise.} \end{cases}$$

Thus, in iterative learning, the learner has access to the previous conjecture, but not to all previous data as in **G**-learning.

Another interesting sequence generating operator is set-driven learning ([28], denoted **Sd**). We let, for all learners h and texts T,

$$\mathbf{Sd}(h, T)(i) = h(\text{content}(T[i])).$$

Successful learning requires the learner to observe certain restrictions, for example convergence to a correct index. These restrictions are formalized in our next definition.

A *sequence acceptance criterion* is a predicate δ on a learning sequence and a text. We give the examples of explanatory (**Ex**) and consistent (**Cons**, [1]) learning, which were discussed in Section 1, as well as conservative learning (**Conv**, [1]). Formally, we let, for all conjecture sequences p and texts T,

$$\mathbf{Ex}(p, T) \Leftrightarrow [\exists q : p \text{ converges to } q \wedge \text{content}(T) = W_q];$$

$$\mathbf{Cons}(p, T) \Leftrightarrow [\forall i : \text{content}(T[i]) \subseteq W_{p(i)}];$$

$$\mathbf{Conv}(p, T) \Leftrightarrow [\forall i : \text{content}(T[i+1]) \subseteq W_{p(i)} \Rightarrow p(i) = p(i+1)].$$

We combine any two sequence acceptance criteria δ and δ' by intersecting them; we denote this by juxtaposition (for example, **Cons** is meant to be always used together with **Ex**). We are also interested in the following relaxation of the **Ex** restriction called **Fex**-learning [8,25,4]. Given $a, b \in \mathbb{N}$, we let $\mathbf{Fex}_b^a$ be the restriction that, after finitely many conjectures, there are only b many conjectures in the remaining learning sequence, and all of them are correct up to a mistakes (incorrect classifications by a conjecture). Clearly, **Ex** is the case of $a = 0$ and $b = 1$. We furthermore allow $a = *$ or $b = *$, to denote an arbitrary *but finite* number. For $\mathbf{Fex}_1^a$ we also sometimes write $\mathbf{Ex}^a$.

Next we introduce several variants of *monotone* learning. The first definition of monotone learning is due to Jantke [13], in the context of function learning. For language learning, monotonicity was first studied in [21]. We define the following sequence acceptance criteria for variants of monotone learning. We let, for all conjecture sequences p and texts T,

$$\mathbf{SMon}(p, T) \Leftrightarrow [\forall i, j : i < j \Rightarrow W_{p(i)} \subseteq W_{p(j)}];$$

$$\mathbf{Mon}(p, T) \Leftrightarrow [\forall i, j : i < j \Rightarrow W_{p(i)} \cap \text{content}(T) \subseteq W_{p(j)} \cap \text{content}(T)];$$

$$\mathbf{WMon}(p, T) \Leftrightarrow [\forall i, j : i < j \wedge \text{content}(T[j]) \subseteq W_{p(i)} \Rightarrow W_{p(i)} \subseteq W_{p(j)}].$$

For any sequence generating operator β and any combination of sequence acceptance restrictions δ, $\mathbf{Txt}\beta\delta$ is a *learning criterion*. A learner h $\mathbf{Txt}\beta\delta$*-learns* the set

$$\mathbf{Txt}\beta\delta(h) = \{L \in \mathcal{E} \mid \forall T \in \mathbf{Txt}(L) : \delta(\beta(h, T), T)\}.$$

Abusing notation, we also use $\mathbf{Txt}\beta\delta$ to denote the set of all $\mathbf{Txt}\beta\delta$-learnable classes (learnable by some learner). For a set $\mathcal{C}$ of learners and a learning criterion $\mathcal{I}$ we write $\mathcal{C}\mathcal{I}$ to restrict the learning criterion to allow only learners from $\mathcal{C}$ for learning.

To make the definitions from the introduction more formal, we say that a learning criterion I *separates* from a learning criterion I' iff there is an $\mathcal{P}I$-learnable set $\mathcal{L}$ which is not $\mathcal{P}I'$-learnable (separation is thus with respect to computable

[5] $h(\emptyset)$ denotes the *initial conjecture* made by h.

learners). I *separates topologically* from a I' iff there is a uniformly linear-time decidable[6] set $\mathcal{L}$ which is $\mathbf{LinF}I$-learnable, but not $\mathfrak{R}I'$-learnable. Thus, topological separation implies separation. We say that I *separates computationally* from I' iff I and I' separate, but *not* topologically.

If all $\mathcal{P}I'$-learnable sets are $\mathcal{P}I$-learnable, but I separates topologically from I', then we denote this very strong separation by $I' \subset_{\text{topo}} I$.

3. Topological constraints

In this section we collect some well-known topological constraints on learning. We start with some strong and important characterizations of $\mathfrak{R}$**TxtGEx**-learning, followed by two more topological restrictions, including the famous theorem about locking sequences (Theorem 3.3, introduced in [2]).

For the characterization of $\mathfrak{R}$**TxtGEx**-learning, we use the following two definitions. A learner is called *prudent* iff it only makes conjectures for languages to be learned [24]. A learner h is called *optimal* iff there is no other learner h' so that (i) h' never converges to a correct conjecture later than h on any text for a language to be learned, and (ii) there is a text for a language to be learned on which h' converges strictly earlier [10].

Theorem 3.1. *Let $\mathcal{L} \subseteq \mathcal{E}$. The following are equivalent.*

(1) $\mathcal{L} \in \mathfrak{R}$**TxtGEx**.
(2) *For each $L \in \mathcal{L}$ there is a finite set $D_L \subseteq L$ such that for all $L' \in \mathcal{L}$ we have that $D_L \subseteq L' \subseteq L$ implies $L' = L$.*
(3) *Let h be the function mapping a given finite sequence of numbers σ to the least index e such that W_e is $\subseteq$-minimal in $\mathcal{L}$ with content$(\sigma) \subseteq W_e$ (the least index for content(σ), if no such e exists). Then h $\mathfrak{R}$**TxtGEx**-learns $\mathcal{L}$.*
(4) *$\mathcal{L}$ is prudently $\mathfrak{R}$**TxtSdConsConvEx**-learnable.*
(5) *$\mathcal{L}$ is optimally $\mathfrak{R}$**TxtSdEx**-learnable.*

Proof. The equivalence of (1) and (2) is known as Angluin's Criterion; Angluin [1] used an effective variant of this to characterize learnability of uniformly computable sets. The version stated here is due to [24], but see also [15,17]. We have "(2) $\Rightarrow$ (3)" directly, as well as "(3) $\Rightarrow$ (4)". The implication "(4) $\Rightarrow$ (5)" was shown in [10] (with a slightly weaker condition and a stronger result); finally, the implication "(5) $\Rightarrow$ (1)" is trivial. □

Note that Theorem 3.1 also implies that **TxtGEx** and **TxtSdEx** do not separate topologically. The finite sets D_L which exist for a given L with respect to a learnable set $\mathcal{L}$ of languages as given by Theorem 3.1, part (2), are called *telltales*.

Another known implication of $\mathfrak{R}$**TxtGEx**-learnability is given by the following proposition and was essentially already known to Gold [11] (this is basically in the statement that no set of languages containing all finite languages and at least one infinite language can be **TxtGEx**-learned).

Proposition 3.2. *Let $\mathcal{L}$ be a set of language. If $\mathcal{L}$ is $\mathfrak{R}$**TxtGEx**-learnable, then, for each infinitely ascending chain $(L_i)_{i\in\mathbb{N}}$, we have $\bigcup_{i\in\mathbb{N}} L_i \notin \mathcal{L}$. Furthermore, the converse does not hold.*

Proof. Let $(L_i)_{i\in\mathbb{N}}$ be an infinitely ascending chain and suppose there is $L \in \mathcal{L}$ such that $\bigcup_{i\in\mathbb{N}} L_i \subseteq L$. It suffices to show that there is $x \in L \setminus \bigcup_{i\in\mathbb{N}} L_i$. From Theorem 3.1 we know that there is D such that $D \subseteq L$ and, for all $i \in \mathbb{N}$, $D \not\subseteq L_i$. Thus, there is $x \in D \setminus \bigcup_{i\in\mathbb{N}} L_i$.

To show that the converse does not hold, consider the set of all co-singletons, together with $\mathbb{N}$. It is well-known that this set is not learnable ($\mathbb{N}$ does not have a finite telltale), but there are no infinitely ascending chains. □

A sequence σ is called a *locking sequence for h on L* iff content$(\sigma) \subseteq L$, $h(\sigma)$ is an index for L and for all ρ with content$(\rho) \subseteq L$ we have $h(\sigma\rho) = h(\sigma)$ [2].

The following well known theorem is probably the most frequent use of topological restrictions to learning.

Theorem 3.3. *(See [2].) Let L be* **TxtGEx***-learned by a learner h. Then, for each sequence σ with content$(\sigma) \subseteq L$ there is τ with content$(\tau) \subseteq L$ such that $\sigma\tau$ is a locking sequence for h on L.*

Note that this generalizes trivially to the case of iterative learning, and also to cases of **Fex**-learning (see [4]). Note that, for any a, b, a $\mathbf{Fex}^a_b$-locking sequence of a learner h on a language L would be a sequence σ of elements from L such that there is a finite set D with $|D| \leq b$ of conjectures correct up to a mistakes such that, for each τ with elements from L, $h(\sigma\tau) \in D$. That is, D is fixed regardless of extension τ.

[6] A set is *uniformly linear-time decidable* iff there is an enumeration $(L_i)_{i\in\mathbb{N}}$ of all the sets from $\mathcal{L}$ such that $\lambda i, x.x \in L_i$ is computable in linear time.

Finally, there are similar characterizations also about strongly monotone learning; for characterizations of computable strongly monotone learning, see [31]. Below we give a characterization for the variant without computability requirements. Note that strongly monotone learning has many regularities (see also [12]).

Theorem 3.4. *Let $\mathcal{L}' \subseteq \mathcal{E}$; let $\mathcal{L}$ be the set of all of $\mathcal{L}'$ and, for each finite set D, the set $\bigcap\{L \in \mathcal{L}' \mid D \subseteq L\}$, where $\bigcap \emptyset = \mathbb{N}$. Then, for each finite set D, there is a minimum $L \in \mathcal{L}$ with $D \subseteq L$ and the following are equivalent.*

(1) *$\mathcal{L}$ is $\mathfrak{R}$**TxtGSMonEx**-learnable.*
(2) *$\mathcal{L}'$ is $\mathfrak{R}$**TxtGSMonEx**-learnable.*
(3) *For each $L \in \mathcal{L}$, there is a finite $D \subseteq \mathbb{N}$ such that L is the minimum element of $\mathcal{L}$ with $D \subseteq L$.*
(4) *Let h be the function mapping a given finite set D to the least index e such that W_e is the minimum element in $\mathcal{L}$ with $D \subseteq W_e$. Then h $\mathfrak{R}$**TxtGSMonEx**-learns $\mathcal{L}$.*

Proof. For each finite set D, $\bigcap\{L \in \mathcal{L}' \mid D \subseteq L\}$ is the minimum $L \in \mathcal{L}$ with $D \subseteq L$.

We now show "(2) $\Rightarrow$ (3)", all other implications (going down and looping around) are straightforward. Let h be a $\mathfrak{R}$**TxtGSMonEx**-learner for $\mathcal{L}'$. It suffices to show that each element $L \in \mathcal{L}'$ has a finite set D as required, as all other members of $\mathcal{L}$ have such a set by definition. Let $L \in \mathcal{L}'$ and let σ be a locking sequence of h on L and let $D = \text{content}(\sigma)$. From $W_{h(\sigma)} = L$ and h strong monotone we know that for each $L' \in \mathcal{L}'$ with $D \subseteq L'$, $L \subseteq L'$. This gives that L is the minimum language in $\mathcal{L}'$ containing D; by the definition of $\mathcal{L}$, this implies that L is also the minimum language of $\mathcal{L}$ containing D. □

4. Topological separations

In this section we present our new topological separations (and non-separations) concerning four different areas of learning criteria. We start with Fex-learning, followed by monotone learning. Third we consider iterative learning followed by a special variant of iterative learning where a learner can query for past data (feedback learning).

4.1. Fex learning

For Fex-learning we get topological separations in the hierarchy concerning the number of mistakes allowed (Theorem 4.1; this is already implicit in [3, Theorem 2]) (see also [4] for a computational proof). However, in contrast to the situation for computable learners, the hierarchy in the maximal number of distinct conjectures collapses to its first level, for all error bounds (Theorem 4.2).

We will use the following notation in this section. For any $a \in \mathbb{N}$ and languages L, L', we write $L =^a L'$ iff the symmetric difference of L and L' contains at most a elements; similarly, we write $L =^* L'$ iff the symmetric difference of L and L' is a finite set.

Theorem 4.1. *(See [3, Theorem 2].) For all a, we have that* $\mathbf{TxtGFex}_1^{a+1}$ *and* $\mathbf{TxtGFex}_*^a$ *separate topologically.*

Proof. Let $\mathcal{L}^{a+1} = \{L \mid L =^{a+1} \mathbb{N}\}$. Clearly, $\mathcal{L}^{a+1} \in \mathbf{LinFTxtGFex}_1^{a+1}$ by a learner which constantly outputs a fixed index for $\mathbb{N}$.

Suppose, by way of contradiction, $\mathcal{L}^{a+1}$ is $\mathfrak{R}\mathbf{TxtGFex}_*^a$-learnable as witnessed by some learner $h \in \mathfrak{R}$. Let σ be a locking sequence for h on $\mathbb{N}$ (in the sense that, after σ, only conjectures for finite variants of $\mathbb{N}$ are presented). Choose any $a+1$ elements not in σ which are contained in all the (finitely many) languages conjectured by h on sequences extending σ. Let L be the set of natural numbers except these $a+1$ elements. Then, clearly, $L \in \mathcal{L}^{a+1}$. However, on any text for L extending σ, h will not output a conjecture for a language with at most a mistakes, a contradiction. □

Theorem 4.2. *For all a, b, we have that*

$$\mathfrak{R}\mathbf{TxtGFex}_1^a = \mathfrak{R}\mathbf{TxtGFex}_b^a.$$

In particular, $\mathbf{TxtGFex}_b^a$ *and* $\mathbf{TxtGFex}_1^a$ *do* not *separate topologically.*

Proof. The inclusion "$\subseteq$" is trivial. For the converse, let $\mathcal{L} \in \mathbf{TxtGFex}_b^a$ as witnessed by $h \in \mathfrak{R}$. We now define some uncomputable functions. Let $D \in \mathfrak{R}$ be such that, for all σ,

$$D(\sigma) = \{h(\tau) \mid \tau \subseteq \sigma \wedge W_{h(\tau)} =^* W_{h(\sigma)}\}.^7$$

Let $h' \in \mathfrak{R}$ be such that, for all σ, $h'(\sigma)$ is minimal with

[7] We use "$\tau \subseteq \sigma$" to denote that σ extends τ.

$$W_{h'(\sigma)} = \text{content}(\sigma) \cup \bigcap_{e \in D(\sigma)} W_e.$$

Let $L \in \mathcal{L}$ and T a text for L. Then D on T converges to a finite set of indices which are finite variants of L (as h only outputs finitely many syntactically distinct conjectures which are eventually all finite variants of L). Let D_0 be this set; note that some conjectures in D_0 may make more than a mistakes. However, D_0 contains some element e_0 with $W_{e_0} =^a L$. We now have, for all i large enough,

$$W_{h'(T[i])} = \text{content}(T[i]) \cup \bigcap_{e \in D_0} W_e.$$

As $\bigcap_{e \in D_0} W_e$ is a finite variant of L, h' on T converges to an index e_1 for a finite variant of L with $L \subseteq W_{e_1}$. Furthermore, we have $W_{e_1} \subseteq L \cup W_{e_0}$, which shows $W_{e_1} =^a L$ as desired. □

4.2. Monotone learning

In this section we want to take a look at some variants of *monotone learning*. Theorem 4.3 gives the results on what separates topologically and what does not.

Theorem 4.3. *We have*

(1) $\mathfrak{R}$**GWMonEx** = $\mathfrak{R}$**GEx**; *and*
(2) **GSMonEx** $\subset_{\text{topo}}$ **GMonEx** $\subset_{\text{topo}}$ **GEx**.

Proof. Regarding (1), the direction "$\subseteq$" is trivial; for the direction "$\supseteq$" we use the characterization given in Theorem 3.1 to see that $\mathcal{L}$ is conservatively learnable, which implies learnable by a weakly monotone learner.

Regarding **GMonEx** $\subset_{\text{topo}}$ **GEx**, consider the set of all co-singleton languages. These are clearly **LinFGEx**-learnable by the learner which conjectures the language which misses only the least not-presented number. Suppose, by way of contradiction, that this set of languages is $\mathfrak{R}$**GMonEx**-learnable, as witnessed by some learner $h \in \mathfrak{R}$. Let σ be a locking sequence of h on $\mathbb{N} \setminus \{0\}$. Let a be the least element of $\mathbb{N}^+ \setminus \text{content}(\sigma)$, and let σ' be such that $\sigma\sigma'$ is a locking sequence for h on $\mathbb{N} \setminus \{a\}$. Then we have $a \in W_{h(\sigma)}$ and $a \notin W_{h(\sigma\sigma')}$, but, for all a' not in $\text{content}(\sigma\sigma')$, $\sigma\sigma'$ can be extended to a text for $\mathbb{N} \setminus \{a'\}$, a contradiction to h monotone.

Regarding **GSMonEx** $\subset_{\text{topo}}$ **GMonEx**, consider $\mathcal{L}$ to contain the set of all even numbers $2\mathbb{N}$, as well as, for each $a \in \mathbb{N}$, the language $L_a = \{2a+1\} \cup \{2b \mid b \leq a\}$. $\mathcal{L}$ can be **LinFGMonEx**-learned by conjecturing $2\mathbb{N}$ until an odd number $2a+1$ is presented, at which time L_a is conjectured. Suppose, by way of contradiction, $\mathcal{L}$ can be $\mathfrak{R}$**GSMonEx**-learned, as witnessed by some learner $h \in \mathfrak{R}$. Let σ be a locking sequence of h on $2\mathbb{N}$. Let a be such that $2a$ is the maximal element in σ (without loss of generality, σ contains at least one element). Then σ can be extended to a text for L_a; thus, let σ' be such that $\sigma\sigma'$ is a locking sequence for h on L_a. Thus, we have $W_{h(\sigma)} = 2\mathbb{N}$, but $2\mathbb{N} \not\subseteq W_{h(\sigma\sigma')}$, a contradiction. □

4.3. Iterative learning

Iterative learning requires the learner to forget past data; thus, it is not surprising that many separations involving iterative learning are topological in nature. We first repeat the well-known proof that iterative learning is less powerful than **TxtGEx**-learning from the introduction [20,22], followed by the topological separation of iterative and consistent iterative learning. In Theorem 4.6 we show that, interestingly, strong non-U-shapedness (and thus conservativeness) and iterative learning separate topologically; this is a significant strengthening of a result from [6], where the separation of iterative and strongly non-U-shaped iterative learning was shown. Furthermore, in Theorem 4.7, we show that *iterative-with-counter* learning separates topologically from iterative learning, strengthening a result from [9]. Finally, we consider coding tricks in iterative learning.

Theorem 4.4. *We have* **TxtItEx** $\subset_{\text{topo}}$ **TxtGEx**.

Proof. Let $\mathcal{L}$ be the set containing $\mathbb{N}^+$ as well as every finite language containing 0. This set of languages is clearly **TxtGEx**-learnable in linear time. Suppose, by way of contradiction, $\mathcal{L}$ is $\mathfrak{R}$**TxtItEx**-learned by a learner $h \in \mathfrak{R}$. Let σ be a locking sequence for h on $\mathbb{N}^+$. Let x and y be two elements from $\mathbb{N}^+ \setminus \text{content}(\sigma)$. Then the sequences $\sigma x 0^\infty$ and $\sigma y 0^\infty$ are for two different languages to be learned by h, but h will converge to the same index on both (if any). □

Theorem 4.5. *We have* **TxtItConsEx** $\subset_{\text{topo}}$ **TxtItEx**.

Proof. Let $\mathcal{L}$ contain the language $L_* = \{\langle a, b\rangle \mid a > 0, b \in \mathbb{N}\}$ as well as, for each $c \in \mathbb{N}$, the language $L_c = \{\langle a, b\rangle \mid a \leq c, b \in \mathbb{N}\}$. Intuitively, as long as no pair $\langle 0, b\rangle$ was encountered, L_* should be guessed; as soon as such a pair is presented, an

iterative learner finds a pair with the largest first component a in the remaining text and updates its conjecture to be for L_a; this can clearly be done in linear time. There will always be a pair with the appropriate a still coming up, as there are infinitely many such pairs in any target language. This shows $\mathcal{L} \in$ **LinFTxtItEx**.

Suppose now, by way of contradiction, $\mathcal{L} \in \mathfrak{R}$**TxtItConsEx** as witnessed by some learner $h \in \mathfrak{R}$. Let σ be a locking sequence for h on L_*. Let $c = \max\{a \mid \exists b : \langle a, b\rangle \in \text{content}(\sigma)\}$. Let σ' be such that $\sigma\sigma'$ is a locking sequence for h on L_c. Then h on $\sigma\langle c+1, 0\rangle$ must output the same conjecture as on σ. Therefore, as h is iterative, h gives the same output on $\sigma\sigma'$ and $\sigma\langle c+1, 0\rangle\sigma'$, a conjecture for L_c by the choice of σ'. Thus, the conjecture is inconsistent on $\sigma\langle c+1, 0\rangle\sigma'$, a sequence which can be extended to a text for L_{c+1}, a contradiction. □

A significant weakening of the restriction of conservativeness is strongly non-U-shapedness [30,10], formally defined as follows. For all conjecture sequences p and texts T,

$$\textbf{SNU}(p, T) \Leftrightarrow [\forall i : \text{content}(T) = W_{p(i)} \Rightarrow p(i) = p(i+1)].$$

Theorem 4.6. *We have* **TxtItSNUEx** $\subset_{\text{topo}}$ **TxtItEx**.

Proof. For this proof, we assume $\langle\cdot,\cdot\rangle$ to be a linear time, linear time invertible pairing function. We define the following sequences, which will serve us as data.

$$\forall i : a(i) = 2i;$$
$$\forall i, k : b_k(i) = 2\langle k, i\rangle + 1.$$

Let $\mathcal{L}$ contain the following languages (we write, for all i, j, $i \equiv_2 j$ iff i and j are both even or both odd, i.e. if they are congruent modulo 2).

$$L_0 = \text{content}(a);$$
$$L_1 = \text{content}(a) \cup \bigcup_{k\in\mathbb{N}} \text{content}(b_k);$$
$$\forall k, j : L_{k,j} = \text{content}(a[k]) \cup \text{content}(b_k[j+1]) \cup \{a(k+i) \mid i < j, i \equiv_2 j\}.$$

Clearly, $\mathcal{L}$ is uniformly linear time decidable with the given numbering. $\mathcal{L}$ is iteratively learnable (in linear time) as follows. First conjecture L_0. As soon as some $b_k(j)$ appeared, memorize k and the maximal j that appears (the k is always the same) and conjecture $L_{k,j}$. Also memorize if ever an $a(k+i)$ with i odd appeared, and also if an $a(k+i)$ with i even appeared. If ever both an $a(k+i)$ with i odd and $a(k+i)$ with i even appeared, conjecture L_1 and stay with this conjecture from now on. As all memorizations can only finitely often lead to a mind change, the resulting learner **TxtItEx**-learns $\mathcal{L}$. Intuitively, the memorizations to learn whether both even and odd data appears is necessary, but forbidden to **SNU**-learners, which we will exploit to get the separation.

Suppose now, by way of contradiction, there is $h \in \mathfrak{R}$ **TxtItSNUEx**-learning $\mathcal{L}$. Let σ be a locking sequence of h on L_0 and let k be minimal such that $\text{content}(\sigma) \subseteq \text{content}(a[k])$. For each j, let τ_j be a listing of all the (finitely many) elements from $\text{content}(a) \cap L_{k,j}$.

We will now recursively define, for all j, a number $z(j) > 0$ as follows. Let j be given such that, for all $i < j$, $z(i) > 0$ is defined. Then

$$\sigma\tau_j b_k(0)^{z(0)} \ldots b_k(j-1)^{z(j-1)} b_k(j)^{\infty}$$

is a text for $L_{k,j}$. As this is a language which is learned by h, there exists $y > 0$ such that h on

$$\sigma\tau_j b_k(0)^{z(0)} \ldots b_k(j-1)^{z(j-1)} b_k(j)^{y}$$

is an index for $L_{k,j}$. Let $z(j) > 0$ be the minimal such y and let

$$\alpha_{j+1} = \sigma\tau_j b_k(0)^{z(0)} \ldots b_k(j)^{z(j)}.$$

We now recursively construct a text T for L_1 as follows. We let

$$\rho_0 = \sigma;$$
$$\forall j : \rho_{j+1} = \rho_j\, b_k(j)^{z(j)}\, \tau_j.$$

We let T be the limit of the ρ_i, i.e., $T = \bigcup_{j\in\mathbb{N}} \rho_j$ (we consider a sequence as the set of pairs mapping an i to the ith element of the sequence). Clearly, T is a text for L_1. Let $\alpha_0 = \sigma$.

We conclude the proof by showing that, for all j, h gives the same output on α_j as on ρ_j, as this shows that h on T makes infinitely many mind changes, a contradiction to h learning L_1. Let h^* be such that, for each finite sequence γ, $h^*(\gamma)$ is the output of h after seeing γ (note that we cannot just write $h(\gamma)$, as h is an iterative learner and expects a previous conjecture and a new datum as input).

Claim. *For all* j, $h^*(\alpha_j) = h^*(\sigma b_k(0)^{z(0)} \dots b_k(j-1)^{z(j-1)}) = h^*(\rho_j)$.

We show the claim by induction on j. The case of $j = 0$ is trivial, as $\alpha_0 = \sigma = \rho_0$. Let now $j > 0$ be such that the claim holds for j.

We have, for all i, $h^*(\sigma\tau_i) = h^*(\sigma)$, as σ is a locking sequence of h on L_0; hence, as h is iterative, we get for all γ $h^*(\sigma\tau_i\gamma) = h^*(\sigma\gamma)$. In particular,

$$h^*(\alpha_{j+1}) = h^*(\sigma\tau_j b_k(0)^{z(0)} \dots b_k(j)^{z(j)}) = h^*(\sigma b_k(0)^{z(0)} \dots b_k(j)^{z(j)}),$$

which gives the first claimed equation. From h iterative and $h^*(\alpha_j) = h^*(\rho_j)$ we know that $h^*(\alpha_j b_k(j)^{z(j)}) = h^*(\rho_j b_k(j)^{z(j)})$ is an index for $L_{k,j}$. We have content$(\tau_j) \subseteq L_{k,j}$; thus, as h learns $L_{k,j}$ strongly non-U-shapedly, h cannot change its mind on data from τ_j when the current conjecture is for $L_{k,j}$, which shows

$$h^*(\rho_j b_k(j)^{z(j)}) = h^*(\rho_j b_k(j)^{z(j)}\tau_j) = h^*(\rho_{j+1})$$

as desired to finish the claim and thus the proof. □

Next we consider a strengthening of iterative learning by giving the learner access to the current iteration number; this is modeled as a new sequence generating operator **ItCtr** (*iterative-with-counter* learning) and was introduced in [9]. Formally we let, for all learners h, texts T and all i,

$$\mathbf{ItCtr}(h, T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0^{8}; \\ h(\mathbf{It}(h, T)(i-1), T(i-1), i-1), & \text{otherwise.} \end{cases}$$

It is known that **TxtSdEx** and **TxtItCtrEx** are incomparable and in between **TxtItEx** and **TxtGEx** [9], and, thus, **TxtItCtrEx** separates from **TxtItEx**. With the next theorem we show that **TxtItCtrEx** separates topologically from **TxtItEx**.

Theorem 4.7. *We have* **TxtItEx** $\subset_{\text{topo}}$ **TxtItCtrEx**.

Proof. For this proof, we again assume $\langle\cdot,\cdot\rangle$ to be a linear time, linear time invertible pairing function. We use again the following sequences as data.

$$\forall i : a(i) = 2i + 1;$$
$$\forall i, j, k : b_k(j) = 2\langle k, j\rangle + 2.$$

Note that 0 is not listed by the functions above. Let $\mathcal{L}$ contain the following languages. For all $k, j > 0$,

$$L_0 = \text{content}(a);$$
$$L_{1,k} = \text{content}(a[k]) \cup \text{content}(b_k);$$
$$L_{2,k,j} = \text{content}(a[k]) \cup \{a(k + \langle i, j\rangle) \mid i \leq j\} \cup \text{content}(b_k[j]) \cup \{0\}.$$

Clearly, $\mathcal{L}$ is uniformly linear time decidable with the given numbering. We fix a (linear time computable) function f such that, for any finite set D and any k,

$$f(k, D) = \max(\{0\} \cup \{j \mid \exists i : a(k + \langle i, j\rangle) \in D\})$$

$\mathcal{L}$ is iteratively-with-counter learnable (in linear time) as shown by the learner given in Algorithm 1. This learner scans through a given text T in a single pass and makes use of the counter (in line 8); otherwise it only stores some additional information in variables, which is possible in iterative-with-counter learning, successful **Ex**-convergence requires that the values of these variables stop changing at some point.

Clearly, this learner uses only linear time in each iteration. It is trivially successful on L_0. For any k, it is successful on $L_{1,k}$ as the variable j is updated at most c_0 times, the variable D is updated at most k times, and the correct conjecture is chosen.

Finally, for any k and j, the learner is successful on any text T for $L_{2,k,j}$ as follows. Let c be minimal such that $T(c) \in (\{0\} \cup \text{range}(b_k))$.

Case 1: $T(c) = 0$. Then the learner will wait for an element $b_k(j')$ to identify the correct k (and storing it). The correct j is found no later than when $b_k(j-1) \in L_{2,k,j}$ is presented; from that point on the conjectures are correct, and D is only updated finitely often.

[8] Recall that $h(\emptyset)$ is the initial hypothesis of h before any data was presented.

Algorithm 1: ItCtr-learner for the proof of Theorem 4.7.

```
Given: Text T;
k ← −1; j ← 0; D ← ∅;
phase ← A;
for c = 0 to ∞ do
    if ∃k′, j′ : T(c) = b_{k′}(j′) and phase = A then
        k ← k′;
        j ← j′;
        c_0 ← c;
        phase ← B;
    if T(c) = 0 then phase ← C;
    if phase = A then conjecture L_0;
    else
        if ∃x : T(c) = a(x) then D ← D ∪ {x};
        if k ≠ −1 then j ← max(j, f(k, D));
        if phase = B then
            if j ≤ c_0 and ∃j′ : T(c) = b_k(j′) then j ← max(j, j′ + 1);
            conjecture L_{1,k};
        else
            if ∃k′, j′ : T(c) = b_{k′}(j′) then
                if k = −1 then k ← k′;
                j ← max(j, j′ + 1);
            if k ≠ −1 then conjecture L_{2,k,j}
```

Case 2: For some $j' < j$, $T(c) = b_k(j')$. Then the algorithm knows the correct k and sets its variable c_0 to c. The j is now identified as follows. If there is ever an element $a(k + \langle i, j\rangle)$ added to D, the correct j is identified directly from this. If else *never* such an element is added to D, then all these elements must have been presented as part of the first c elements; this implies $j < c$. The learner will update j in Phase B with each new datum $b_k(j')$ as necessary, as $j' \leq j < c = c_0$. Thus, we now have that the learner updates j at each $b_k(j')$ in the text and finds also in this case the correct value of j when seeing $b_k(j-1)$. Again, D is only updated finitely often.

This shows that $\mathcal{L}$ is **TxtItCtrEx**-learnable in linear time.

Intuitively, the learner which has a counter knows how many of the $b_k(j)$ to use to update its j-value, while an iterative learner without a counter will necessarily fail to memorize sufficiently, as we will now prove formally.

Suppose, by way of contradiction, there is $h \in \mathfrak{R}$ which **TxtItEx**-learns all of $\mathcal{L}$. Let σ be a locking sequence of h on L_0 and let k be minimal such that $\text{content}(\sigma) \subseteq \text{content}(a[k])$. Let τ be such that $\sigma\tau$ is a locking sequence for h on $L_{1,k}$. Let $j_0 > 0$ be minimal such that $\text{content}(\tau) \subseteq \text{content}(a[k]) \cup \text{content}(b_k[j_0])$. For all j, let α_j be a sequence of all elements in $L_{2,k,j} \cap \text{content}(a)$, and β_j be a sequence of all elements in $L_{2,k,j} \cap \text{content}(b_k)$. Consider the following two texts.

$$\sigma\, \alpha_{j_0+2}\, \tau\, \beta_{j_0+2}\, 0^\infty;$$
$$\sigma\, \alpha_{j_0+3}\, \tau\, \beta_{j_0+3}\, 0^\infty.$$

These are texts for L_{2,k,j_0+2} and L_{2,k,j_0+3}, respectively. However, from the chosen locking sequences and h being an iterative learner, we have that h converges to the same conjecture on both of these texts, a contradiction. □

In [16] the authors investigate the interesting question of how much *coding* helps with iterative learning. Loosely speaking, *coding* refers to an iterative learner exploiting the access to the current conjecture for storage purposes, by coding the information to be stored into the conjecture. The authors defined and analyzed a very interesting collection of learning criteria which aim at restricting the ability to exploit such coding. Here we just want to mention two of these learning criteria. One of the most restricted criteria requires the learner to exclusively use hypotheses from a *Friedberg numbering*, a complete and effective numbering of all computably enumerable sets, *without repetitions*. A much more relaxed learning criterion called extensional **TxtItEx** allows using the W-system for conjectures; however, it is required that, when presented with equivalent conjecture and identical input elements, the learner must produce equivalent conjectures.

It is easy to see that these two restrictions do not separate topologically, but that in fact they allow for learning the same sets of languages by learners from $\mathfrak{R}$. However, the separation of **TxtItEx** and extensional **TxtItEx** shown in [16] makes use only of topological arguments and a very simple set of languages, so that we get the following theorem.

Theorem 4.8. *(See [16, Theorem 22].)* **TxtItEx** *and extensional* **TxtItEx** *separate topologically.*

Furthermore, it is easy to see that the set from [16] witnessing the topological separation can be modified to contain infinite languages only.

4.4. Feedback learning

There are many extensions of iterative learning studied in the literature. In this section we are interested in *feedback* learning, where a learner is allowed to query for past data [29,22]. In particular, we are interested in hierarchies spanned by feedback learners [5].

We will model feedback learning with up to $k \in \mathbb{N}$ (parallel) feedback queries as a specific sequence generating operator $\mathbf{Fb}_k$. The learner has the same information in each iteration as in iterative learning, but can first choose a set of up k elements and then use the additional information of which of these elements have been presented before to compute the next conjecture. Formally, the sequence generating operators are defined as follows. Let $k \in \mathbb{N}$. For any finite set D we let $\min_k(D)$ denote the least k elements of D. For all learners h and texts T we define

$$f(h,T)(i) = \operatorname{content}(T[i-1]) \cap \min_k(h(0, \mathbf{Fb}_k(h,T)(i-1), T(i-1)));$$

$$\mathbf{Fb}_k(h,T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0; \\ h(1, \mathbf{Fb}_k(h,T)(i-1), T(i-1), f(h,T)(i)), & \text{otherwise.} \end{cases}$$

In this definition, h on any arguments starting with 0 returns a set of data items to be recalled; f returns the set of successfully recalled data items. h on arguments starting with 1 returns the new conjecture.

The first theorem will show that, in general, the separations in the hierarchy of feedback learning are witnessed by topological separations. However, unlike for computable learners (see [5]), when restricted to sets of infinite languages only, the hierarchy collapses to its first layer (Theorem 4.11).

First we note that the hierarchy holds in general also topologically.

Theorem 4.9. *For all $k > 0$, we have that* $\mathbf{TxtFb}_k\mathbf{Ex}$ *and* $\mathbf{TxtFb}_{k-1}\mathbf{Ex}^*$ *separate topologically. In particular,* $\mathbf{TxtFb}_{k-1}\mathbf{Ex} \subset_{\text{topo}} \mathbf{TxtFb}_k\mathbf{Ex}$.

Proof. Let $k \in \mathbb{N}$. For each $i < k$ and each x, let $a_i(x) = 2(kx+i)$. Note that $\lambda i, x.a_i(x)$ is 1–1 with range $2\mathbb{N}$ (with i ranging over natural numbers $< k$). For each t, x let $b_t(x) = 2\langle t, x\rangle + 1$. We have $\lambda t, x.b_t(x)$ is 1–1 with range $2\mathbb{N}+1$.

Let $\mathcal{L}$ contain the following languages. For all $j < k$ and all $t, y \in \mathbb{N}$,

$$\tilde{L} = 2\mathbb{N};$$
$$L_t = \bigcup_{i<k} \{a_i(x) \mid x < t\} \cup \operatorname{range}(b_t);$$
$$L_{j,t,y} = \bigcup_{i<k} \{a_i(x) \mid x < t\} \cup \{b_t(x) \mid x \le y\} \cup \{a_j(t+y)\}.$$

We have $\mathcal{L} \in \mathbf{TxtFb}_k\mathbf{Ex}$ by a learner h_0 as follows. The initial conjecture is for $\tilde{L}$. When an element $b_t(x)$ is presented, query for $\{a_i(t+x) \mid i < k\}$. If none of the queries is positive, conjecture and index for L_t; if $a_j(t+x)$ is positive, conjecture an index for $L_{j,t,x}$ (no other results are consistent with $\mathcal{L}$) and keep this conjecture henceforth. If the current conjecture is for L_t and $a_j(x)$ is presented for some $x \ge t$, output a conjecture for $L_{j,t,x-t}$ and never change. In all other cases, do not change the conjecture. It is straightforward to verify that this learner will $\mathbf{TxtFb}_k\mathbf{Ex}$-learn $\mathcal{L}$.

Suppose, by way of contradiction, $\mathcal{L} \in \mathbf{TxtFb}_{k-1}\mathbf{Ex}^*$ as witnessed by some learner h. Let σ be a locking sequence for h on $\tilde{L}$ (the final conjecture will, thus, be for a finite variant of $\tilde{L}$). Let t be such that $\operatorname{content}(\sigma) \subseteq \bigcup_{i<k}\{a_i(x) \mid x < t\}$. Let τ be a sequence of all elements in $\bigcup_{i<k}\{a_i(x) \mid x < t\}$. Consider the text $\sigma\tau b_t$ for L_t. Then there are $j < k$ and $y \in \mathbb{N}$ such that (i) after $\sigma\tau b_t[y]$, the conjecture is for a finite variant of L_t; (ii) h is converged on $\sigma\tau b_t$ after $\sigma\tau b_t[y]$; and (iii) h, while being presented the data $\sigma\tau b_t[y+1]$, never queried $a_j(t+y)$. These j and y exist as only $k-1$ queries are allowed per iteration, but k more items $a_j(t+y)$ are possible after each iteration.

Now we have that h on the text

$$\sigma\tau a_j(t+y) b_t[y] b_t(y)^\infty$$

for $L_{j,t,y}$ will converge to the same conjecture as on $\sigma\tau b_t$, a contradiction (the two languages are not finite variants). □

The class witnessing the separation in the just prior proof employed finite as well as infinite languages. Already in [5] it was noted that the hierarchy collapses to the first level when concerned with sets of infinite languages only, if the class to be learned can be indexed such that membership is uniformly decidable. We will generalize this result by showing more generally how, given *any text for an infinite language from a countable set*, one can extract a listing where each natural numbers occurs infinitely often. Such a listing in learning was termed an *onto counter* in [19] (see also [9]) and, with the help of a single feedback query, can be used to simulate *fat text*, a text where each datum is presented infinitely often (see [17, Proposition 3.37]); the details can be found in the proof of Theorem 4.11.

Lemma 4.10. *Let $\mathcal{L}$ be a countable set of infinite languages. Then there is a function $f : \mathbb{N} \to \mathbb{N}$ such that, for each $L \in \mathcal{L}$, f restricted to L has infinitely many pre-images for each element of $\mathbb{N}$. Furthermore, if $\mathcal{L}$ is uniformly computably enumerable,[9] then f is computable.*

Proof. Let $(L_i)_{i\in\mathbb{N}}$ be an enumeration of the elements of $\mathcal{L}$. We define the function f inductively via a growing set D of pairs $\langle x, y\rangle$; we will define f as mapping any such x to its associated y.

In the formal argument we will use a function g mapping any finite set of pairs D and an i to the minimum element $x \in L_i$ which is not the left part of a pair in D (for the "furthermore" clause we will choose the first $x \in L_i$ found which is larger than any left part of a pair in D). This always exists, as all L_i are supposed infinite and there are only finitely many pairs in D. We let

$$D_0 = \emptyset;$$
$$D_{\langle i,y,t\rangle+1} = D_{\langle i,y,t\rangle} \cup \{\langle g(D_{\langle i,y,t\rangle}, i), y\rangle\}.$$

Let $D = \bigcup_{i\in\mathbb{N}} D_i$. It is clear that for each x there is at most one y with $\langle x, y\rangle \in D$. Thus, D is the graph of a (partial) function; let f be an arbitrary extension of this function.

To show the correctness, let $L \in \mathcal{L}$ and let i be such that $L_i = L$; let $y \in \mathbb{N}$. For all t, from the definition of $D_{\langle i,y,t\rangle+1}$ we see that there is an $x_t \in L_i$ such that $\langle x_t, y\rangle \in D$. Thus, for all t, $f(x_t) = y$ as desired.

The "furthermore" clause is straightforward. □

The idea is now as follows. When trying to learn a set of infinite languages $\mathcal{L}$, we can use an associated f to produce a fat text as follows: if presented with a data from an $L \in \mathcal{L}$, mapping all data with f will enumerate all of $\mathbb{N}$ infinitely often. This effectively produces an onto counter. Using these numbers for the queries will allow for learning iteratively with fat text, which is known to equal **TxtGEx**-learning [17, Proposition 3.37].

See also [5, Theorem 5] for a similar theorem, specialized to uniformly computable sets of infinite languages, but with stronger conclusion, which can be concluded with the "furthermore" clause in Lemma 4.10 and Angluin's original telltale condition for uniformly decidable sets of languages [1].

Theorem 4.11. *Let $\mathcal{L}$ be a set of infinite computably enumerable languages. Then*

$$\mathcal{L} \in \mathfrak{R}\mathbf{TxtFb}_1\mathbf{Ex} \Leftrightarrow \mathcal{L} \in \mathfrak{R}\mathbf{TxtGEx}.$$

In particular, **TxtGEx** *and* $\mathbf{TxtFb}_1\mathbf{Ex}$ *do* not *separate topologically* on sets of infinite languages.

Proof. The implication "⇒" is trivial. For the converse, let $\mathcal{L} \in \mathbf{TxtGEx}$.

Let f be a function as given by Lemma 4.10 for $\mathcal{L}$ (or even all infinite computably enumerable sets). Let $(L_i)_{i\in\mathbb{N}}$ be an enumeration of $\mathcal{L}$. According to the characterization from Theorem 3.1, we know that $\mathcal{L}$ is learnable via telltale sets. We let p be a 1–1 function such that $p(D)$ is an index for L_i with i minimal such that L_i contains all of D and the telltale for L_i is contained in D (an index for $\emptyset$, if no such i exists).

Next we define a 1-feedback learner $h \in \mathfrak{R}$. For this we note that a 1-feedback learner can make arbitrarily many feedback queries m at the cost of changing its conjecture for m iterations (ignoring the new input data). Thus, whenever a learner makes a mind change anyway, arbitrarily much data can be queried.

The initial hypothesis of h is $p(\emptyset)$. When presented with datum z and previous hypothesis $p(D)$, query for $f(z)$. If the query comes out negative or $f(z)$ is already included in the language corresponding to the current hypothesis, h keeps its old conjecture $p(D)$. Otherwise, suppose the current conjecture of h is for some L_j; then h makes a mind change and, using additional iterations as described above, h queries *all the data* of all the (finite) telltales of the languages $(L_i)_{i<j}$, gathering the positives in a set D'. The conjecture of h is now $p(D \cup D' \cup \{f(z)\})$. Using f, h effectively simulates iterative learning from *fat text*, a text where each datum is presented infinitely often; this is known to equal $\mathfrak{R}$**TxtGEx**-learning [17, Proposition 3.37].

Regarding correctness, let $L \in \mathcal{L}$ and let T be a text for L. From the choice of f we know that h queries each item infinitely often. In particular, every L' with $L \setminus L' \neq \emptyset$ previous in the order of $(L_i)_{i\in\mathbb{N}}$ will be discarded eventually. As soon as all the elements from the telltale of L have been presented, these will be queried at the next mind change, after which the conjecture will stay correct. It follows from the telltale condition that no incorrect conjecture can be kept indefinitely. □

5. Conclusion

In this paper we have seen that separation proofs in inductive inference can have computational components, or are completely topological. We gave a number of examples for both from diverse areas of inductive inference and also showed instances where no topological separations exist. We hope that this line of research contributes to our understanding of (a)

[9] A set of languages $\mathcal{L}$ is *uniformly computably enumerable* iff there is $r \in \mathcal{R}$ such that $\mathcal{L} = \{W_{r(i)} \mid i \in \mathbb{N}\}$.

the structure of learning and (b) the proofs employed for showing separations. Future publications could use the concepts introduced in this paper to help readers understand a proof by mentioning whether it is computational or topological.

For topological restrictions, we have required decision procedures for witnessing classes to be computable in linear time, the same for the learners of these classes. It is an interesting open question whether, for reasonable learning criteria, we get the same learning criteria separating topologically if we only required, for example, total computability—our examples do not give a counterexample to such a claim.

Finally, we only gave a glimpse of how the topology of learnable classes is restricted. To go deeper, it might be worth investigating Baire category theory [14] in more explicit detail.

Acknowledgements

The authors would like to thank the anonymous reviewers both of the conference version and of the journal version, which helped improve the current version.

References

[1] Dana Angluin, Inductive inference of formal languages from positive data, Inform. Control 45 (2) (May 1980) 117–135.
[2] Leonore Blum, Manuel Blum, Toward a mathematical theory of inductive inference, Inform. Control 28 (2) (June 1975) 125–155.
[3] Ganesh Baliga, John Case, Learnability: admissible, co-finite, and hypersimple languages, J. Comput. System Sci. 53 (1) (1996) 26–32, first appeared in ICALP '93.
[4] John Case, The power of vacillation in language learning, SIAM J. Comput. 28 (6) (1999) 1941–1969.
[5] John Case, Sanjay Jain, Steffen Lange, Thomas Zeugmann, Incremental concept learning for bounded data mining, Inform. and Comput. 152 (1) (1999) 74–110.
[6] John Case, Timo Kötzing, Strongly non-U-shaped learning results by general techniques, in: Adam Tauman Kalai, Mehryar Mohri (Eds.), Proceedings of the Learning Theory, 23rd Conference on Learning Theory, COLT 2010, Haifa, Israel, June 27–29, 2010, Omnipress, 2010, pp. 181–193.
[7] John Case, Timo Kötzing, Topological separations in inductive inference, in: Sanjay Jain, Rémi Munos, Frank Stephan, Thomas Zeugmann (Eds.), Proceedings of the Algorithmic Learning Theory, 24th International Conference, ALT 2013, Singapore, October 6–9, 2013, in: Lecture Notes in Artificial Intelligence, vol. 8139, Springer, Berlin, Heidelberg, New York, 2013, pp. 128–142.
[8] John Case, Christopher Lynes, Machine inductive inference and language identification, in: Automata, Languages and Programming, 9th Colloquium, Proceedings, vol. 140, Springer-Verlag, 1982, pp. 107–115.
[9] John Case, Samuel E. Moelius III, *U*-shaped, iterative, and iterative-with-counter learning, in: Proceedings of the Learning Theory, 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA, June 13–15, 2007, in: Lecture Notes in Artificial Intelligence, vol. 4539, Springer, Berlin, 2007, pp. 172–186.
[10] John Case, Samuel E. Moelius III, Optimal language learning, in: Proceedings of the Algorithmic Learning Theory, 19th International Conference, ALT 2008, Budapest, Hungary, October 2008, in: Lecture Notes in Artificial Intelligence, vol. 5254, Springer, Berlin, October 2008, pp. 419–433.
[11] E. Mark Gold, Language identification in the limit, Inform. Control 10 (5) (1967) 447–474.
[12] Jeffrey Heinz, Anna Kasprzik, Timo Kötzing, Learning in the limit with lattice-structured hypothesis spaces, Theoret. Comput. Sci. 457 (1) (2012) 111–127.
[13] Klaus P. Jantke, Monotonic and non-monotonic inductive inference, New Gener. Comput. 8 (4) (1991) 349–360.
[14] Thomas Jech, Set Theory, Academic Press, NY, 1978.
[15] Dick De Jongh, Makoto Kanazawa, Angluin's theorem for indexed families of r.e. sets and applications, in: Proceedings of the Computational Learning Theory, Proceedings of the Ninth Annual Conference, COLT 1996, Desenzano del Garda, Italy, June 28–July 1, 1996, ACM, 1996, pp. 193–204.
[16] Sanjay Jain, Samuel E. Moelius III, Sandra Zilles, Learning without coding, in: Special Issue on Learning Theory, Theoret. Comput. Sci. 473 (2013) 124–148.
[17] Sanjay Jain, Daniel Osherson, James S. Royer, Arun Sharma, Systems that Learn: An Introduction to Learning Theory, second edition, MIT Press, Cambridge, Massachusetts, 1999.
[18] Timo Kötzing, Abstraction and complexity in computational learning in the limit, PhD thesis, University of Delaware, 2009, available online at http://pqdtopen.proquest.com/#viewpdf?dispub=3373055.
[19] Timo Kötzing, Iterative learning from positive data and counters, in: Jyrki Kivinen, Csaba Szepesvári, Esko Ukkonen, Thomas Zeugmann (Eds.), Proceedings of the Algorithmic Learning Theory, 22nd International Conference, ALT 2011, Aalto University, Espoo, Finland, October 5–7, 2011, in: Lecture Notes in Artificial Intelligence, vol. 6925, Springer, Berlin, Heidelberg, New York, 2011, pp. 40–54.
[20] Efim Kinber, Frank Stephan, Language learning from texts: mindchanges, limited memory, and monotonicity, Inform. and Comput. 123 (2) (1995) 224–241.
[21] Steffen Lange, Thomas Zeugmann, Monotonic versus non-monotonic language learning, in: Nonmonotonic and Inductive Logic, Second International Workshop, Reinhardsbrunn Castle, Germany, December 1991, in: Lecture Notes in Artificial Intelligence, vol. 659, Springer-Verlag, 1993, pp. 254–269.
[22] Steffen Lange, Thomas Zeugmann, Incremental learning from positive data, J. Comput. System Sci. 53 (1) (1996) 88–103.
[23] Daniel N. Osherson, Michael Stob, Scott Weinstein, Note on a central lemma of learning theory, J. Math. Psych. 27 (1983) 86–92.
[24] Daniel N. Osherson, Michael Stob, Scott Weinstein, Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists, MIT Press, Cambridge, Massachusetts, 1986.
[25] Daniel N. Osherson, Scott Weinstein, Criteria of language learning, Inform. Control 52 (2) (1982) 123–138.
[26] James Royer, John Case, Subrecursive Programming Systems: Complexity and Succinctness, Research Monograph in Progress in Theoretical Computer Science, Birkhäuser, Boston, 1994.
[27] Hartley Rogers, Theory of Recursive Functions and Effective Computability, McGraw Hill, New York, 1967. Reprinted by MIT Press, Cambridge, Massachusetts, 1987.
[28] Kenneth Wexler, Peter W. Culicover, Formal Principles of Language Acquisition, MIT Press, Cambridge, Massachusetts, 1980.
[29] Rolf Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Elektron. Inf.verarb. Kybern. 12 (1/2) (1976) 93–99.
[30] Rolf Wiehagen, A thesis in inductive inference, in: Proceedings of the Nonmonotonic and Inductive Logic, 1st International Workshop, Karlsruhe, Germany, December 1990, in: Lecture Notes in Artificial Intelligence, vol. 543, Springer-Verlag, 1990, pp. 184–207.
[31] Thomas Zeugmann, Steffen Lange, Shyam Kapur, Characterizations of monotonic and dual monotonic language learning, Inform. and Comput. 120 (2) (1995) 155–173.