Topological Separations in Inductive Inference

John Case[1] and Timo Kötzing[2,*]

[1] Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716-2586, USA
case@cis.udel.edu

[2] Institut für Informatik, Jena University, Germany
timo.koetzing@uni-jena.de

Abstract. A major question asked by learning in the limit from positive data is about what classes of languages are learnable with respect to a given learning criterion. We are particularly interested in the reasons for a class of languages to be *un*learnable. We consider two types of reasons. One type is called *topological* (as an example, Gold has shown that no class containing an infinite language and all its finite sub-languages is learnable). Another reason is called *computational* (as the learners are required to be algorithmic). In particular, two learning criteria might allow for learning different classes of languages because of different topological restrictions, or because of different computational restrictions.

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 Fex, monotone, 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 *some* computational restrictions.

1 Introduction

The learning theory of this paper pertains to trial and error learning of descriptions, i.e., grammars or programs, for formal languages L. This kind of learning is sometimes called *learning in the limit*, and herein it's learning from positive data only re such L. The languages are taken 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 deciding whether a learner h is *successful* on a language L have been proposed in the literature. Gold, in his seminal paper [Gol67], gave

* The authors would like to thank the anonymous reviewers for their valuable feedback.

a first, simple learning criterion, we call **Txt GEx**-*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 **Txt GEx**). Trivially, each single, describable language L has a suitable constant function as an **Txt GEx**-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 **Txt GEx**-learning have been investigated (see, for example, the textbook [JORS99]).

Already Gold [Gol67] found that certain classes of languages are not **Txt GEx**-learnable because of what was later called topological considerations[2], e.g., when trying to **Txt GEx**-learn a set 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). Angluin [Ang80] described another essentially topological restriction of **Txt GEx**-learning. Intuitively, when one of these restrictions is not met, the learner just does not get enough information to be successful, regardless of its computational power. We collect a number of previously known topological constraints on **Txt GEx**-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 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 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 **Txt GEx**-learning as **Txt GConsEx** when only computable learners are considered. It is well known that **Txt GEx** separates from **Txt GConsEx** [OSW86]. An example set that cannot be **Txt GEx**-learned consistently is the set of all non-empty languages where the least element is a coded

[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 [Gol67]; *Ex* stands for *explanatory*.

[2] These topological considerations arise, for example, for **Txt GEx**-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 *un*computable hs. The associated proofs of unlearnability typically feature directly or indirectly winning moves in a Banach-Mazur game where the goal set is co-meager — as in Baire category theory [Jec78] — and Baire category theory is part of topology. The connection to Baire category theory was first observed in [OSW83] (see also [OSW86]).

description for the language (a code in some acceptable numbering of all computably enumerable sets). It is clear that the only reason that this set 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 [Wie76, WC80]. 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 this learning criterion is called **TxtItEx**. It is well-known that **TxtGEx** separates from **TxtItEx**. We consider the following proof of this separation [KS95, LZ96]. Let $\mathcal{L}$ be the set containing the language $\mathbb{N}^+$ (the language of all positive natural numbers) as well as every finite language containing 0. This set of languages is clearly **TxtGEx**-learnable, even by learners which map a string of inputs to a conjecture in linear time. However, this set cannot be **TxtItEx**-learned. For suppose, by way of contradiction, that some (possibly even non-computable) h would **TxtItEx**-learn this set of languages. Then, when being fed positive numbers, h will eventually output a conjecture for $\mathbb{N}^+$ and not change any more. If now, after some more positive numbers, a 0 is presented, h has "forgotten" what more positive numbers were presented. A more formal proof can be found after the statement of Theorem 8 below. This shows how iterative learning leaves the learner at an informational disadvantage; even removing any requirement of computability for the learner cannot enable the iterative learning.

Intuitively, we want to call separations as in 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 set 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). Note that the restriction to linear time is somewhat arbitrary; in both cases this restriction is present to make sure that really only topological properties witness the separation. A further advantage is that we the separations we get are stronger than if we would require only uniform computability. It is an interesting open question whether, for reasonable learning criteria, this restriction to linearly computable languages makes a difference. 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, because testing consistency is an uncomputable task (in general one 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 9 in Section 4 below shows. We next summarize informally some of our other main theorems also in Section 4 below.

For $k > 0$, $\mathbf{TxtGFex}_k$-learning is just like **TxtGEx**-learning except that instead of being restricted to exactly 1 correct output conjecture in the limit, $\mathbf{TxtGFex}_k$-learning allows up to k correct output conjectures in the limit. Computationally, i.e., with algorithmic learners h, from [Cas99], these criteria form a strict learning power hierarchy with increasing k; however, surprisingly, from our Theorem 6 below, topological separation fails, and the hierarchy collapses when uncomputable learners are also allowed.

$\mathbf{TxtFb}_k\mathbf{Ex}$-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 [CJLZ99] it is shown that these criteria also form a strict learning power hierarchy *computationally* with increasing k; however, surprisingly, from our Theorem 12 below, the hierarchy *also holds topologically.* But, from our Theorem 13 below, the hierarchy *collapses topologically* when the potential separation witnesses $\mathcal{L}$ are restricted to contain *no* finite languages.

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 computable power.

Note that some proofs are not included because of space restrictions.

2 Mathematical Preliminaries

Unintroduced complexity theoretic notation follows [RC94]. Other unintroduced notation follows [Rog67].

$\mathbb{N}$ 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. For any set A, we let $\mathrm{Pow}(A)$ denote the set of all subsets of A. $\emptyset$ 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 $\lambda x_1, \ldots, x_n \centerdot f(x_1, \ldots, x_n)$. For example, with $c \in \mathbb{N}$, $\lambda x \centerdot c$ is the constantly c function of one argument.

We let $\langle \cdot, \cdot \rangle$ be a linear time computable, linear time invertible, pairing function [RC94] (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., $\exists x_0 : f(x_0) = p \wedge \forall x \geq x_0 : f(x)\downarrow \in \{?, p\}$.[3] $\mathcal{P}$ 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).

2.1 Learning Criteria

In this section we formally introduce our setting of learning in the limit and associated learning criteria. We follow [Köt09] 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 ce 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 $\mathrm{content}(T) = L$. 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.

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

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,

$$\mathbf{G}(h,T)(i) = h(T[i]);$$
$$\mathbf{It}(h,T)(i) = \begin{cases} h(\emptyset), & \text{if } i = 0;^{4} \\ h(\mathbf{It}(h,T)(i-1), T(i-1)), & \text{otherwise.} \end{cases}$$

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 (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**, [Ang80]) learning, which were discussed in Section 1, as well as conservative learning (**Conv**, [Ang80]). Formally, we let, for all conjecture sequences p and texts T,

$$\begin{aligned} \mathbf{Ex}(p,T) &\Leftrightarrow [\exists q : p \text{ converges to } q \wedge \text{content}(T) = W_q]; \\ \mathbf{Cons}(p,T) &\Leftrightarrow [\forall i : \text{content}(T[i]) \subseteq W_{p(i)}]; \\ \mathbf{Conv}(p,T) &\Leftrightarrow [\forall i : \text{content}(T[i+1]) \subseteq W_{p(i)} \Rightarrow p(i) = p(i+1)]. \end{aligned}$$

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 [CL82, OW82, Cas99]. 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 [Jan91], in the context of function learning. For language learning, monotonicity was first studied in [LZ93]. We define the following sequence acceptance criteria for variants of monotone learning.

$$\begin{aligned} \mathbf{SMon} &= \{(p,T) \mid \forall i,j : i < j \Rightarrow W_{p(i)} \subseteq W_{p(j)}\}; \\ \mathbf{Mon} &= \{(p,T) \mid \forall i,j : i < j \Rightarrow W_{p(i)} \cap \text{content}(T) \subseteq W_{p(j)} \cap \text{content}(T)\}; \\ \mathbf{WMon} &= \{(p,T) \mid \forall i,j : i < j \wedge \text{content}(T[j]) \subseteq W_{p(i)} \Rightarrow W_{p(i)} \subseteq W_{p(j)}\}. \end{aligned}$$

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

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 learners). I *separates topologically* from a I' iff there is a uniformly linear-time decidable[5] 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 $I' \subset_{\mathrm{topo}} I$.

3 Topological Constraints

In this section we collect some well-known topological constraints on learning. We start with some strong and important characterizations of $\mathfrak{R}\mathbf{Txt}\mathbf{GEx}$-learning, followed by two more topological restrictions, including the famous theorem about locking sequences (Theorem 3, introduced in [BB75]).

Theorem 1. Let $\mathcal{L} \subseteq \mathcal{E}$. The following are equivalent.

(1) $\mathcal{L} \in \mathfrak{R}\mathbf{Txt}\mathbf{GEx}$.
(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 $\mathrm{content}(\sigma) \subseteq W_e$ (the least index for $\mathrm{content}(\sigma)$, if no such e exists). Then h $\mathfrak{R}\mathbf{Txt}\mathbf{GEx}$-learns $\mathcal{L}$.
(4) $\mathcal{L}$ is prudently $\mathfrak{R}\mathbf{Txt}\mathbf{SdConsConvEx}$-learnable.[6]
(5) $\mathcal{L}$ is optimally $\mathfrak{R}\mathbf{Txt}\mathbf{SdEx}$-learnable.[7]

Proof. The equivalence of (1) and (2) is known as Angluin's Criterion; Angluin [Ang80] used an effective variant of this to characterize learnability of uniformly computable sets. The version stated here is due to [OSW86], but see also [dJK96,

[5] 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 $\lambda i, x \centerdot x \in L_i$ is computable in linear time.

[6] *Prudence* refers to a learner making only conjectures for languages to be learned.

[7] *Optimal* language learning was discussed in [CM11] and refers to learners which could not converge earlier on a given text for a language to be learned without converging later on another text for a language to be learned.

JORS99]. We have "(2) $\Rightarrow$ (3)" directly, as well as "(3) $\Rightarrow$ (4)". The implication "(4) $\Rightarrow$ (5)" was shown in [CM11] (with a slightly weaker condition and a stronger result); finally, the implication "(5) $\Rightarrow$ (1)" is trivial. $\square$

Note that Theorem 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 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 [Gol67] (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 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 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. $\square$

A sequence σ is called a *locking sequence for h on L* iff $\text{content}(\sigma) \subseteq L$, $h(\sigma)$ is an index for L and for all ρ with $\text{content}(\rho) \subseteq L$ we have $h(\sigma\,\rho) = h(\sigma)$ [BB75].

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

Theorem 3 ([BB75]). Let L be **TxtGEx**-learned by a learner h. Then, for each sequence σ with $\text{content}(\sigma) \subseteq L$ there is τ with $\text{content}(\tau) \subseteq L$ 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 [Cas99]).

Finally, we can make similar characterizations also about strongly monotone learning, which has many regularities (see also [HKK12]).

Theorem 4. Let $\mathcal{L}' \subseteq \mathcal{E}$; let $\mathcal{L}$ contain 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 $\mathcal{L}$.

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 5; this is already implicit in [BC96, Theorem 2]) (see also [Cas99] 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 6).

Theorem 5 ([BC96, Theorem 2]). For all a, we have that $\mathbf{Txt\,GFex}_1^{a+1}$ and $\mathbf{Txt\,GFex}_*^{a}$ separate topologically.

Theorem 6. For all a, b, we have that

$$\mathfrak{R}\mathbf{Txt\,GFex}_1^a = \mathfrak{R}\mathbf{Txt\,GFex}_b^a.$$

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

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

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

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

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

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,

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

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. $\square$

4.2 Monotone Learning

In this section we want to take a look at some variants of *monotone learning.* Theorem 7 gives the results on what separates topologically and what does not.

Theorem 7. We have

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

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

Regarding $\mathbf{GMonEx} \subset_{\text{topo}} \mathbf{GEx}$, 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 $\mathfrak{R}\mathbf{GMonEx}$-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 content$(\sigma\sigma')$, $\sigma\sigma'$ can be extended to a text for $\mathbb{N} \setminus \{a'\}$, a contradiction to h monotone.

Regarding $\mathbf{GSMonEx} \subset_{\text{topo}} \mathbf{GMonEx}$, 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 $\mathfrak{R}\mathbf{GSMonEx}$-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 [KS95, LZ96], followed by the topological separation of iterative and consistent iterative learning (we omit the proof due to space constraints). Finally, we consider coding tricks in iterative learning.

Theorem 8. We have $\mathbf{TxtItEx} \subset_{\text{topo}} \mathbf{TxtGEx}$.

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 $h \in \mathfrak{R}$. We let h^* be the **TxtGEx**-learner equivalent to h. 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 9. We have **TxtItConsEx** $\subset_{\text{topo}}$ **TxtItEx**.

In [JMZ13] 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 [JMZ13] makes use only of topological arguments and a very simple set of languages, so that we get the following theorem.

Theorem 10 ([JMZ13, Theorem 22]). **TxtItEx** and extensional **TxtItEx** separate topologically.

Furthermore, it is easy to see that the set from [JMZ13] 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 [Wie76, LZ96]. In particular, we are interested in hierarchies spanned by feedback learners [CJLZ99].

We will model feedback learning with upto $k \in \mathbb{N}$ (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.

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 [CJLZ99]), when restricted to sets of infinite languages only, the hierarchy collapses to its first layer (Theorem 13). First we note that the hierarchy holds in general also topologically.

Theorem 11. 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 $\lambda i, x\,\boldsymbol{.}\,a_i(x)$ 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 $\lambda t, x\,\boldsymbol{.}\,b_t(x)$ is $1-1$ with range $2\mathbb{N} + 1$.

Let $\mathcal{L}$ contain the following languages.

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

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 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 item are possible 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). $\square$

The class witnessing the separation in the just prior proof employed finite as well as infinite languages. Already in [CJLZ99] 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 of all natural numbers infinitely often. Such a listing in learning was termed an *onto counter* in [Köt11] (see also [CM08]) 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 [JORS99, Proposition 3.37]); the details can be found in the proof of Theorem 13.

Lemma 12. 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,[8] 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

$$D_0 = \emptyset; \tag{1}$$

$$D_{\langle i,y,t\rangle+1} = D_{\langle i,y,t\rangle} \cup \{\langle g(D_{\langle i,y,t\rangle}, i), y\rangle\}. \tag{2}$$

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 [JORS99, Proposition 3.37].

See also [CJLZ99, 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 12 and Angluin's original telltale condition for uniformly decidable sets of languages [Ang80].

Theorem 13. Let $\mathcal{L}$ be a set of infinite computably enumerable languages. Then

$$\mathcal{L} \in \mathfrak{R}\mathbf{TxtFb_1Ex} \Leftrightarrow \mathcal{L} \in \mathfrak{R}\mathbf{TxtGEx}.$$

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

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

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

Let f be a function as given by Lemma 12 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 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 [JORS99, 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. ☐

References

[Ang80] Angluin, D.: Inductive inference of formal languages from positive data. Information and Control 45, 117–135 (1980)

[BB75] Blum, L., Blum, M.: Toward a mathematical theory of inductive inference. Information and Control 28, 125–155 (1975)

[BC96] Baliga, G., Case, J.: Learnability: Admissible, co-finite, and hypersimple sets. Journal of Computer and System Sciences 53, 26–32 (1996)

[Cas99] Case, J.: The power of vacillation in language learning. SIAM Journal on Computing 28, 1941–1969 (1999)

[CJLZ99] Case, J., Jain, S., Lange, S., Zeugmann, T.: Incremental concept learning for bounded data mining. Information and Computation 152, 74–110 (1999)

[CL82] Case, J., Lynes, C.: Machine inductive inference and language identification. In: Nielsen, M., Schmidt, E.M. (eds.) ICALP 1982. LNCS, vol. 140, pp. 107–115. Springer, Heidelberg (1982)

[CM08] Case, J., Moelius, S.: U-shaped, iterative, and iterative-with-counter learning. Machine Learning 72, 63–88 (2008)

[CM11] Case, J., Moelius, S.: Optimal language learning from positive data. Information and Computation 209, 1293–1311 (2011)

[dJK96] de Jongh, D., Kanazawa, M.: Angluin's thoerem for indexed families of r.e. sets and applications. In: Proc. of COLT (Computational Learning Theory), pp. 193–204 (1996)

[Gol67] Gold, E.: Language identification in the limit. Information and Control 10, 447–474 (1967)

[HKK12] Heinz, J., Kasprzik, A., Kötzing, T.: Learning in the limit with lattice-structured hypothesis spaces. Theoretical Computer Science 457, 111–127 (2012)

[Jan91] Jantke, K.: Monotonic and non-monotonic inductive inference of functions and patterns. In: Dix, J., Schmitt, P.H., Jantke, K.P. (eds.) NIL 1990. LNCS, vol. 543, pp. 161–177. Springer, Heidelberg (1991)

[Jec78] Jech, T.: Set Theory. Academic Press, NY (1978)

[JMZ13] Jain, S., Moelius, S., Zilles, S.: Learning without coding. Theoretical Computer Science 473, 124–148 (2013)

[JORS99] Jain, S., Osherson, D., Royer, J., Sharma, A.: Systems that Learn: An Introduction to Learning Theory, 2nd edn. MIT Press, Cambridge (1999)

[Köt09] Kötzing, T.: Abstraction and Complexity in Computational Learning in the Limit. PhD thesis, University of Delaware (2009), `http://pqdtopen.proquest.com/#viewpdf?dispub=3373055`

[Köt11] Kötzing, T.: Iterative learning from positive data and counters. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 40–54. Springer, Heidelberg (2011)

[KS95] Kinber, E., Stephan, F.: Language learning from texts: Mind changes, limited memory and monotonicity. Information and Computation 123, 224–241 (1995)

[LZ93] Lange, S., Zeugmann, T.: Monotonic versus non-monotonic language learning. In: Proc. of Nonmonotonic and Inductive Logic, pp. 254–269 (1993)

[LZ96] Lange, S., Zeugmann, T.: Incremental learning from positive data. Journal of Computer and System Sciences 53, 88–103 (1996)

[OSW83] Osherson, D., Stob, M., Weinstein, S.: Note on a central lemma of learning theory. Journal of Mathematical Psychology 27, 86–92 (1983)

[OSW86] Osherson, D., Stob, M., Weinstein, S.: Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists. MIT Press, Cambridge (1986)

[OW82] Osherson, D., Weinstein, S.: Criteria of language learning. Information and Control 52, 123–138 (1982)

[RC94] Royer, J., Case, J.: Subrecursive Programming Systems: Complexity and Succinctness. Research Monograph in Progress in Theoretical Computer Science. Birkhäuser, Boston (1994)

[Rog67] Rogers, H.: Theory of Recursive Functions and Effective Computability. McGraw Hill, New York (1987); reprinted by MIT Press, Cambridge (1987)

[WC80] Wexler, K., Culicover, P.: Formal Principles of Language Acquisition. MIT Press, Cambridge (1980)

[Wie76] Wiehagen, R.: Limes-Erkennung rekursiver Funktionen durch spezielle Strategien. Elektronische Informationverarbeitung und Kybernetik 12, 93–99 (1976)